PENINGKATAN PERFORMA TRANSFER DATA PADA PC ROUTER DENGAN PENDEKATAN OPTIMASI OPERATING SYSTEM

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

HARMADI WIBOWO

10251020357

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU

2008

PENINGKATAN PERFORMA TRANSFER DATA PADA PC ROUTER DENGAN PENDEKATAN OPTIMASI OPERATING SYSTEM

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

HARMADI WIBOWO
10251020357

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2008

ABSTRACT

Router is a devices that fowarding data packet from one network to another network. PC router is an alternative to replace a router that make a personal computer become identically with characteristic of router. To make a personal computer to be a router, the computer must have two or more interface for upload and download port and then an operating system that supporting for routing.

Operating system that a lot used at PC router is Linux and BSD. Obviously Linux that used in this research (by installation default), have the actual transfer performance result that really not optimum. Therefore the objective of this research is to optimize the PC router in order to make the data transfer performance optimum and efficient.

Because the characteristic of Linux is open source, the writer try to optimize the PC router from the operating system side, which is kernel and service. The changes from MTU (Maximum Transfer Unit) value test, give impact to PC router, which is throughput reduction.

Key word : BSD*, linux ,* MTU*, open source,* PC *router, Router, throughput.*

ABSTRAK

Router merupakan suatu perangkat yang dapat mem-*foward* paket data dari jaringan satu ke jaringan lainnya. PC *router* merupakan salah satu alternatif pengganti *router* yang menjadikan sebuah perangkat *komputer* menjadi indentik dengan karakteristik dari sebuah *router*. Untuk menjadikan perangkat *komputer* menjadi sebuah PC *router*, perangkat *komputer* tersebut harus memiliki dua atau lebih *interface* untuk *port upload* dan *download* serta sistem operasi yang mendukung untuk *routing*.

Sistem operasi yang banyak dipakai pada PC *router* adalah *Linux* dan BSD. Ternyata *Linux* yang digunakan pada penelitian ini dalam keadaan instalasi default, performa transfer data yang dihasilkan belum optimal. Untuk itu penelitian ini bertujuan untuk mengoptimasi PC *router* agar transfer data yang dilakukan menjadi optimal dan efisien.

Karena karakeristik *Linux* yang *open source*, penulis mencoba optimasi dari sisi operating sistem yang ada, yaitu pada *kernel* dan *service*. Perubahan pada nilai MTU (*Maximum Transfer Unit*) yang diujikan ternyata memberikan dampak yaitu penurunan *throughput* pada PC *router*.

Kata kunci : BSD*, linux ,* MTU*, open source,* PC *router, Router, throughput.*

DAFTAR ISI

BAB I

PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi komunikasi semakin lama semakin menuntut kebutuhan pelayanan yang tinggi. *Router* merupakan salah satu teknologi yang sangat membantu dalam hal teknologi komunikasi, terutama dalam hal jaringan atau *network*. *Router* dapat mengirimkan data dari dan ke tempat yang sangat jauh. Bahkan *router* memudahkan untuk berkomunikasi satu dengan lainnya dibelahan dunia lain. *Router* merupakan suatu alat yang berfungsi untuk meneruskan paket dari satu jaringan ke jaringan lain yang berbeda. Tanpa *router*, komunikasi dengan jaringan luar tidak akan dapat terjadi.

Dalam perkembangan *internet, router* memiliki peran yang sangat penting. *Router* meneruskan paket dari *Local Area Network* ke jaringan lainnya yang lebih luas yaitu *Internet*. Perkembangan *router* dan penggunaannya mengalami kemajuan, baik dalam hal teknologi *hardware* maupun protokol. *Router* memiliki kemampuan yang lebih baik daripada *Switch* atau *Hub*, karena *router* mampu melewatkan paket dari satu jaringan ke jaringan yang lain, yang biasanya letaknya atau pemetaannya sangat berjauhan. Untuk itu *router* menggunakan tabel *routing* dan protokol *routing* yang berfungsi untuk mengatur lalu lintas data. Paket yang datang diperiksa dahulu di *router* dan diteruskan ke alamat yang dituju. Agar paket data yang diterima dapat sampai ke tujuan dengan cepat, *router* harus memproses data tersebut dengan sangat cepat menggunakan *Central Processing Unit* (CPU) yang ada dan meneruskannya kembali ke tujuan. Selain itu *router* juga dilengkapi pula dengan beberapa *interface* untuk keluar masuknya data. Kemampuan sebuah *router* biasanya teletak pada *hardware* atau *interface* yang dipakai. Namun kemampuan sebuah *router* tidak hanya terletak pada *hardware* dan *interface* yang dipakai, *Operating System* juga sangat mempengaruhi kinerja dari sebuah *router*. Baik itu *router* buatan pabrik (*dedicated router*) maupun PC

router. *Dedicated router* memiliki *operating system* yang berdiri sendiri, tergantung kepada *vendor* yang membuatnya. Perusahaan *Cisco* misalnya, menanamkan *Internetwork Operating System* (IOS) kepada seluruh *router*-nya. Demikian pula dengan PC *router*, PC *router* juga menggunakan *operating system* untuk merubah fungsi dari sebuah PC menjadi PC *router*.

Biasanya kemampuan *router* buatan pabrik (*dedicated router*) lebih baik daripada PC *router*. Walaupun memiliki kekurangan, PC *router* tetap masih banyak dipakai oleh kalangan menengah kebawah karena faktor ekonomisnya, sedangkan *dedicated router* banyak digunakan pada jaringan *enterprise*. PC *router* juga dapat memanfaatkan PC lama 486 atau PC *Pentium* 1 yang tidak terpakai lagi. Kemampuan transfer data yang pada PC *router* kurang memuaskan bagi *user* sedangkan kapasitas *bandwith maximum* yang didapat juga belum *maximal*. Sehingga hal tersebut yang menjadi latar belakang penelitian ini.

1.2. Rumusan Masalah

Penelitian yang akan dilakukan adalah “bagaimana meningkatkan performa dari PC *router* berbasis Linux dengan cara melakukan optimasi pada *operating system* sehingga mendapatkan performa transfer data yang maksimum”.

1.3. Batasan Masalah

Batasan masalah dalam penelitian ini meliputi:

1. Penelitian dilakukan untuk melihat seberapa jauh optimasi yang dilakukan pada *operating system* PC *router* dapat meningkatkan performa transfer data pada *network.*
2. Parameter pengukuran performa yang digunakan sebagai acuan dalam penelitian ini adalah *Throughput* (*Transfer Rate)* dalam *Kbytes/sec dan Kbits/sec*.
3. *Operating System* yang digunakan adalah *Linux*. Selain bersifat *Open Source*, *Linux* dikenal sebagai *operating system* yang memiliki distribusi yang banyak dan dapat di-*customize* oleh banyak *vendor*.

4. Optimasi *operating system* yang akan dilakukan meliputi:
    a. Optimasi *Kernel*

       Optimasi *kernel* yang dilakukan dalam hal ini meliputi:

       Upgrade kernel dan optimasi dari modul-modul *kernel* yang secara *default* yang termuat ke dalam kernel, dimana hal ini menyebabkan banyak modul yang tidak terpakai di-*load* ke *memory router*. Dengan ukuran dan parameter *kernel* yang *optimum*, maka akan dapat menghemat *resource* sehingga *Throughput* yang akan didapatkan menjadi semakin besar.

    b. *Service*

       Optimasi dari beberapa *service* yang tersedia didalam sebuah *operating system*. Tidak semua *service* yang di-*load* oleh *operating system* dibutuhkan dalam fungsinya sebagai *router*. Identifikasi yang tepat dari *service* yang dibutuhkan akan memperbaiki kinerja dari *router*.

    c. *MTU (Maximum Transfer Unit)*

       Optimasi yang dilakukan juga meliputi pada nilai MTU. Tujuannya adalah untuk mengetahui apakah besar MTU mempengaruhi nilai *throughput* dari transfer data yang kita lakukan.
5. PC yang digunakan sebagai PC *router* adalah PC dengan spesifikasi *Pentium* 233 MMX dengan *RAM* 96MB
6. PC *router* yang digunakan sebagai perbandingan adalah PC dengan spesifikasi *Pentium* 233 MMX dengan *RAM* 192MB dan *Pentium* 3 dengan *RAM* 128MB.
7. Yang digunakan sebagai *port upload* dan *download* pada PC *router* adalah dua buah Fast Ethernet 10/100.
8. Peralatan yang digunakan adalah *un-manageable switch* dan kabel *UTP category* 5 yang berjalan dengan kecepatan 100 Mbps. Tidak menggunakan peralatan yang melebihi batas dari kecepatan diatas, misalnya *Gigabit Ethernet* atau *10BaseT* yang berjalan dibawahnya.

9. Sebagai *client* digunakan 2 buah PC dengan spesifikasi *Pentium* 3 dengan *RAM* 128MB.
10. *Tools* yang digunakan untuk simulasi adalah *iperf, Qcheck, ftp server 3Cdaemon* dan *ftp client*.
11. Tipe data yang digunakan adalah file *multimedia* dan file *document*. Diantaranya adalah file video (*avi)* dan kumpulan file *document (txt, jpg, html, pdf* dan *zip)* yang jumlah dan kapasitasnya ditentukan dan akan di-*download* dan di-*upload* melalui aplikasi *FTP*.
12. Penelitian hanya meliputi untuk satu PC *router* saja. Dalam hal ini *routing* protokol yang ada dalam sebuah *router*, seperti *RIP*, *OSPF*, dan *BGP* tidak akan diterapkan. Untuk itu penelitian akan difokuskan kepada hasil analisa *traffic* data yang didapatkan sebelum dan sesudah dilakukannya optimasi terhadap *operating system*.
13. Penelitian juga membatasi bahwa protokol yang akan diterapkan disini adalah protokol *TCP/IP*, karena protokol *TCP/IP* yang merupakan protokol yang *independent* dan hampir setiap *vendor* maupun *device* yang ada, menerapkan dan *compatible* terhadap protokol TCP/IP yang ada.
14. PC *router* yang telah dioptimasi juga akan dibandingkan dengan PC *router* yang lain dengan *operating system Linux Live CD Router*. Selain itu juga digunakan PC *router* dengan *hardware* yang berbeda sebagai pembanding.

1.4. Tujuan

Penelitian ini bertujuan untuk melihat bagaimana performa dari sebuah PC *router* sebelum dan sesudah dilakukannya optimasi serta bagaimana sebuah PC *router* bekerja menyampaikan informasi dari satu jaringan ke jaringan yang berlainan.

Tujuan dalam penelitian ini adalah sebagai berikut:

1. Untuk melihat seberapa besar performa yang didapat dari hasil penelitian ini berdasarkan hasil *throughput* dari transfer data yang didapat. Sehingga pengguna dapat memaksimalkan kinerja dari PC *router* mereka.
2. Implementasi *Linux* sebagai *operating system* untuk optimasi *PC router*.

3. Menguji performa *operating system Linux* pada *PC router* melalui perubahan nilai MTU (*Maximum Transfer Unit*) pada *Ethernet Card*. Apakah ada perbedaan sebelum dan setelah dioptimasi *operating system*-nya.
4. Optimasi bertujuan untuk membuat *PC router* dapat berkerja optimal tanpa terbebani oleh modul-modul serta *service* yang ter-*load* pada *operating system*, sehingga didapatkan alokasi *resource* yang lebih baik.
5. Membandingkan *operating system Linux* yang berlainan distribusi, sehingga kemampuan *operating system* tersebut dapat dilihat pada PC *router*.

1.6. Metodologi

Dalam penelitian dan pembuatan penulisan yang membahas *Peningkatan Performa Transfer Data Pada PC Router Dengan Pendekatan Optimasi Operating System*, digunakan metode sebagai berikut:

1. Metode Pengumpulan Data
    a. Studi pustaka

       Data didapatkan melalui studi pustaka serta hasil observasi yang dilakukan oleh penulis.

    b. Observasi

       Melakukan pengamatan pada implementasi data yang didapatkan dengan cara melakukan *test* transfer data terhadap PC *router* pada saat sebelum dan setelah dioptimasi serta melakukan transfer data pada PC *Router* pembanding.
2. Analisa
    a. Menganalisa performa *PC router* berdasarkan hasil *throughput* dari *traffic data* yang ada, sebelum dan sesudah dilakukannya optimasi pada *operating system*.
    b. Mencari keuntungan dan kelemahan-kelemahan yang mungkin terjadi dari penelitian tersebut.

3. Implementasi
    a. Perancangan suatu *network* dalam skala kecil dengan menerapkan *protokol* TCP/IP.
    b. PC *router* akan di-*install* oleh *operating system Linux*, sedangkan *client* memakai *operating system* dari *Microsoft.*
    c. Akan dilakukan optimasi terhadap sistem operasi *router*, meliputi optimasi terhadap *kernel* dan *service* serta seting MTU pada ethernet card yang berhubungan dengan *operating system* yang ada.
    d. Hasil *throughput* dari penelitian ini, sebelum dan sesudah dilakukannya *optimasi* serta perbandingannya dengan PC *router* pembanding akan menjadi dasar penelitian dan perbandingan dalam menentukan performa sebuah *PC router*.

1.7. Sistematika Penulisan

Pembahasan sistematika penulisan menjadi 5 bab yang terdiri dari:

BAB 1 PENDAHULUAN

Pendahuluan akan membahas latar belakang, rumusan masalah, batasan masalah, tujuan, metode penelitian dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Landasan teori akan membahas teori-teori yang berhubungan dengan topik penelitiannya, yaitu jaringan komunikasi data, sistem operasi dan *Linux*

BAB 3 ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas analisa dan perancangan network dengan menggunakan PC *router* yang akan diujikan. Terbagi menjadi beberapa sub bab, yaitu analisa fungsi kernel dan service pada *Linux*, pengaruh MTU terhadap transfer data dan perancangan simulasi

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini akan membahas implementasi dan hasil yang didapatkan dari pengujian, terangkum dalam implementasi optimasi PC *router*, batasan optimasi dan simulasi, optimasi PC *router* dan hasil pengujian transfer data pada PC *router*.

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran, merangkum isi yang dibahas pada bab sebelumnya.

BAB II

LANDASAN TEORI

2.1 Jaringan Komunikasi Data

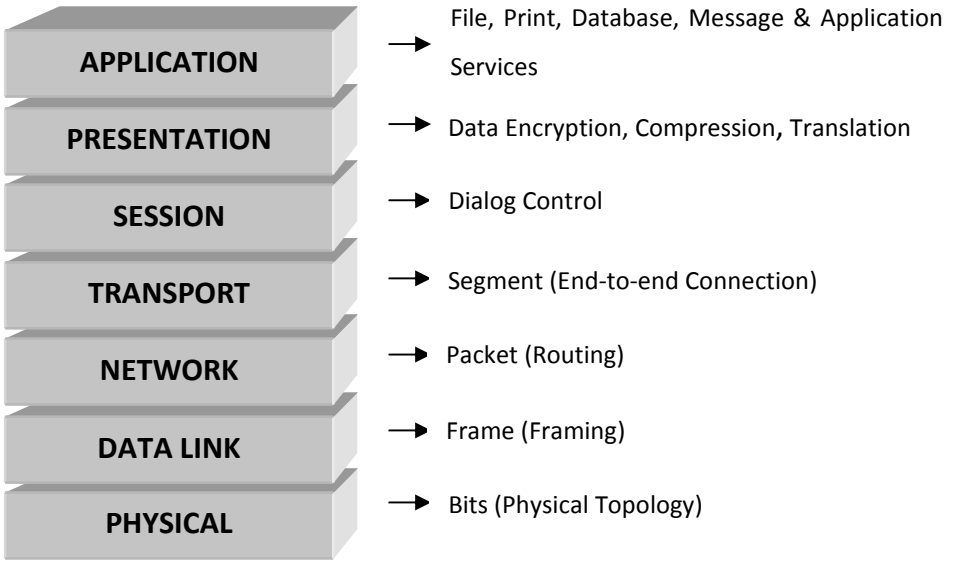
Dengan berkembangnya teknologi *komputer* dan komunikasi suatu model komputer tunggal yang melayani seluruh tugas-tugas komputasi suatu organisasi kini telah diganti dengan sekumpulan *komputer* yang terpisah-pisah akan tetapi saling berhubungan dalam melaksanakan tugasnya, sistem seperti ini disebut jaringan *komputer* (Tanenbaum, 1996).

Dalam prakteknya jaringan *komputer* merupakan sekumpulan *terminal* atau *host* yang terhubung satu sama lain pada lokasi tertentu dengan menggunakan perangkat jaringan dan media fisik, sehingga perangkat tersebut dapat saling berkomunikasi dan berbagi sumber daya dengan menggunakan aturan atau protokol tertentu. *Host* dapat berupa *workstation*, *server*, *router* dan terminal aktif lainnya. Sedangkan perangkat jaringan dapat berupa *Network Card Interface* (*NIC*) dan kabel.

2.1.1 Model Reference OSI

Model referensi OSI (*Open System Interconnection*) menggambarkan bagaimana informasi dari suatu *software* aplikasi di sebuah *komputer* berpindah melewati sebuah media jaringan ke suatu aplikasi di *komputer* lain. Model referensi OSI secara *konseptual* terbagi ke dalam tujuh lapisan dimana masing-masing lapisan memiliki fungsi jaringan yang spesifik. Model ini diciptakan berdasarkan sebuah proposal yang dibuat oleh *International Standards Organization* (ISO) sebagai langkah awal menuju standarisasi protokol internasional yang digunakan pada berbagai *layer*. Model ini disebut *OSI Reference Model* karena model ini ditujukan untuk pengkoneksian *open system*. *Open system* dapat diartikan sebagai suatu sistem yang terbuka untuk

berkomunikasi dengan sistem-sistem lainnya. Berikut adalah gambar dari tujuh lapisan *OSI layer* (Lammle, 2006).



Gambar 2.1. Tujuh Lapisan OSI Layer

Model *OSI* memiliki tujuh *layer*. Prinsip-prinsip yang digunakan bagi ketujuh *layer* tersebut adalah:

1. Sebuah *layer* harus dibuat bila diperlukan tingkat abstraksi yang berbeda.
2. Setiap *layer* harus memiliki fungsi-fungsi tertentu.
3. Fungsi setiap *layer* harus dipilih dengan teliti sesuai dengan ketentuan standar protokol internasional.
4. Batas-batas *layer* diusahakan agar meminimalkan aliran informasi yang melewati *interface*.
5. Jumlah *layer* harus cukup banyak, sehingga fungsi-fungsi yang berbeda tidak perlu disatukan dalam satu *layer* diluar keperluannya. Akan tetapi jumlah *layer* juga harus diusahakan sesedikit mungkin sehingga arsitektur jaringan tidak menjadi sulit dipakai.

Ketujuh *OSI Reference Model* (Lammle, 2006) tersebut adalah:

1. *Application Layer*

   Application layer dari model lapisan *OSI* merupakan lapisan pertama yang berfungsi sebagai sarana komunikasi *user* dan komputer. Lapisan ini berperan ketika komputer membutuhkan akses menuju *network*. Lapisan ini bertanggung jawab untuk mengidentifikasi dan mengadakan komunikasi dengan lawan ketika dibutuhkan dan memastikan apakah *resource* komunikasi yang dibutuhkan tersedia atau tidak. Sebagai contoh *Internet Explorer (IE).* Anda dapat menghapus *service* yang tidak diperlukan oleh *IE*, seperti *TCP/IP, NIC*, *modem*, dan lainnya. Namun anda dapat terus menggunakan *IE* untuk melihat dokumen *HTML* yang ada pada *harddrive* anda. Dan itu tidak menjadi masalah bagi anda. Tapi semuanya akan menjadi masalah jika anda mencoba melihat dokumen *HTML* tersebut melalui *HTTP* atau mengambil file melalui *FTP*. Itu semua karena *IE* merespon untuk permintaan tersebut dengan mencoba akses melalui *application layer*. Permintaan tersebut tentunya tidak akan pernah sampai ke tujuan dikarenakan berhenti pada *layer* yang berada dibawahnya. Dengan kata lain, *IE* sebenarnya bukan merupakan application *layer*. *IE* merupakan *interface* untuk *application layer* protokol ketika komunikasi dengan *remote resources* dibutuhkan.

2. *Presentation Layer*

   Persentation layer mendapatkan namanya dari tujuannya. Ia menghadirkan data kepada *application layer* dan bertanggung jawab atas *data translation* dan *code formating*. *Layer* ini dibutuhkan sebagai penerjemah dan menyediakan *coding* serta fungsi konversi. Keberhasilan teknik penyampaian data yang dihasilkan adalah dengan menyesuaikan data dengan format standar sebelum ditransmisikan. Komputer dibuat untuk menerima format data umum seperti ini dan mengkonversi data tersebut ke dalam format aslinya sehingga dapat dibaca. Sebagai contoh, *EBCDIC* dikonversikan ke *ASCII*.

3. *Session Layer*

   Session layer bertanggung jawab dalam hal menemukan, me-*manage*, dan memutuskan sesi aktif antara *persentation layer*. *Layer* ini juga menyediakan dialog kontrol antar *devices* dan *nodes*. *Session layer* juga mengkordinasi komunikasi antar *system* dan *service* untuk mengatur komunikasi dengan memberikan tawaran terhadap tiga model yang berbeda; *simplex, half duplex,* dan *full duplex*. Secara umum dapat disimpulkan *session layer* menjaga aplikasi data yang berbeda terpisah dari aplikasi data lainnya.

4. *Transport Layer*

   Fungsi utama *transport layer* adalah menerima data dari *session layer* dan meneruskannya ke *network layer* serta menjamin semua data tersebut dapat tiba disisi lainnya dengan benar. *Transport layer* bertanggung jawab untuk menyediakan mekanisme untuk *multiplexing layer* diatasnya, mengadakan *session* dan memutuskan *virtual circuit*. *Transport layer* juga menyembunyikan detail informasi dari *network* pada *layer* diatasnya dengan menyediakan transfer data yang transparant. Contoh protokol yang bekerja pada *transport layer* adalah *TCP* (*Transmission Control Protocol*) dan *UDP* (*User Datagram Protocol*).

5. *Network Layer*

   Network layer mengatur pengalamatan suatu *devices*, *tracking* lokasi dari suatu *devices* dalam sebuah jaringan, dan memutuskan jalur yang terbaik untuk dilewati oleh data, yaitu *network layer* mesti melewatkan aliran data antar *devices* yang tidak terhubung secara lokal. *Router* (*layer 3 devices*) spesifik dengan *network layer* dan menyediakan layanan *routing* dalam *internetwork.*

6. *Data Link Layer*

   Data link layer menyediakan transmisi fisik dari data dan menangani *error notification*, *network topology*, dan *flow control*. *Data link layer* memastikan bahwa pesan yang disampaikan akan dikirim ke-*device* yang

tepat dengan melihat kepada alamat *hardware* (*mac address*) dan menterjemahkan pesan tersebut dari *network layer* ke dalam bilangan *biner* (*bits*) untuk dikirimkan melalui *physical layer*. *Data link layer* memformat pesan yang akan disampaikan menjadi bagian-bagian yang disebut dengan *data frame*. *Switch* merupakan implementasi dari *data link layer*.

7. *Physical Layer*

   Physical layer bertanggung jawab untuk mengirimkan sinyal 1 dan 0 (*biner*) dan meng-*encode* sinyal tersebut ke dalam sinyal digital untuk ditransmisikan ke dalam jaringan.

2.1.2 Internetwork

Tujuan dibuatnya jaringan *komputer* adalah untuk menyampaikan data dari satu *komputer* ke *komputer* yang lain. Namun jika jaringan tidak dapat menyampaikan data ke tujuan yang seharusnya, maka jaringan tersebut menjadi tidak berguna. (Onno, 2001). Dalam komunikasi data, jaringan dapat terhubung melalui satu jaringan ke jaringan lainnya. Semakin lama pertumbuhan jaringan menjadi semakin besar dan akses layanan pun semakin beragam. Kualitas dan performa jaringan dituntut utuk dapat memenuhi kebutuhan konsumen yang kian kompleks. Apabila dua perangkat komunikasi dihubungkan secara langsung dari ujung ke ujung, akan terjadi kemungkinan-kemungkinan sebagai berikut:

1. Bila perangkat-perangkatnya merupakan bagian yang masing-masing jauh terpisah, berada pada jarak ribuan kilometer, tentunya akan memakan biaya yang sangat mahal untuk menyambung dan menghubungkannya
2. Terdapat serangkaian perangkat, masing-masing akan membutuhkan jaringan untuk menghubungkan satu sama lain pada waktu yang berbeda.

Solusi masalah ini adalah dengan cara menghubungkan masing-masing perangkat tersebut dengan suatu jaringan komunikasi (*internetworking*).

2.1.2.1 *Router*

Router memiliki kemampuan melewatkan paket *IP* dari satu jaringan ke jaringan lain yang mungkin memiliki banyak jalur diantara keduanya. *Router-router* yang saling terhubung dalam jaringan internet turut serta dalam sebuah algoritma *routing* terdistribusi untuk menentukan jalur terbaik yang akan dilalui paket IP dari sistem ke sistem lain (Wijaya, 2003). Proses *routing* dilakukan secara *hop by hop*. IP tidak mengetahui jalur keseluruhan menuju tujuan setiap paket. IP *routing* hanya menyediakan IP *address* dari *router* berikutnya yang menurutnya lebih dekat ke host tujuan, dengan mempertimbangkan jalur terbaik untuk dilalui oleh paket-paket IP. *Router* tidak peduli terhadap *host* yang akan menerima paket tersebut, *router* hanya peduli terhadap alamat *network* yang dituju.

Router dapat digunakan untuk menghubungkan sejumlah LAN sehingga trafik yang dibangkitkan oleh suatu LAN terisolasikan dengan baik. Beberapa poin yang harus diingat tentang *router* (Lammle, 2006), yaitu;

1. *Router* secara default tidak mem-*forward* paket *broadcast*.
2. *Router* menggunakan *logical address* pada network *layer header* untuk memastikan *hop* selanjutnya dari *router* sehingga paket dapat diteruskan kembali ke *router* selanjutnya.
3. *Router* dapat menggunakan *Access List*, yang memudahkan pada *administrator* untuk mengontrol tingkat keamanan dari paket untuk keluar masuk dari suatu *interface*.
4. *Router* (*Layer 3 devices*) dapat menyediakan koneksi antar *Virtual LAN* (*VLANs*).

Jika dua atau lebih LAN terhubung dengan *router*, setiap LAN dianggap sebagai *subnetwork* yang berbeda. Mirip dengan *bridge*, *router* dihubungkan *network interface* yang berbeda. *Router* terletak pada *Layer 3* dalam OSI, *router* hanya perlu mengetahui *Network Id* (nomor jaringan) dari data yang diterimanya untuk diteruskan ke jaringan yang dituju. Cara kerjanya setiap paket data yang datang, paket data tersebut dibuka lalu dibaca *header* paket datanya kemudian dicocokkan atau dibandingkan ke dalam *routing table* yang ada dan diteruskan kembali ke jaringan yang dituju melalui suatu *interface*.

2.1.2.2 Protokol

Model OSI menyediakan secara konseptual kerangka kerja untuk komunikasi antar komputer, tetapi model ini bukan merupakan metoda komunikasi. Sebenarnya komunikasi dapat terjadi karena menggunakan protokol komunikasi. Di dalam konteks jaringan data, sebuah protokol adalah suatu aturan formal dan kesepakatan yang menentukan bagaimana komputer bertukar informasi melewati sebuah media jaringan. Sebuah protokol mengimplementasikan salah satu atau lebih dari lapisan-lapisan OSI. Sebuah variasi yang lebar dari adanya protokol komunikasi, tetapi semua memelihara pada salah satu aliran group: protokol LAN, protokol WAN, protokol jaringan, dan protokol *routing*. Protokol LAN beroperasi pada lapisan fisik dan *data link* dari model OSI dan mendefinisikan komunikasi di atas macam-macam media LAN. Protokol WAN beroperasi pada ketiga lapisan terbawah dari model OSI dan mendefinisikan komunikasi di atas macam-macam WAN. Protokol *routing* adalah protokol lapisan jaringan yang bertanggung jawab untuk menentukan jalan dan pengaturan lalu lintas data untuk beberapa *network.* Akhirnya protokol jaringan adalah berbagai protokol dari lapisan teratas yang ada dalam sederetan protokol.

2.2 Sistem Operasi

Sistem operasi merupakan sebuah penghubung antara pengguna dari komputer dengan perangkat keras komputer. Sebelum ada sistem operasi, orang hanya mengunakan komputer dengan menggunakan sinyal *analog* dan sinyal *digital*. Seiring dengan berkembangnya pengetahuan dan teknologi, pada saat ini terdapat berbagai sistem operasi dengan keunggulan masing-masing. Secara umum definisi sistem operasi ialah pengelola seluruh sumber daya (*resource*) yang terdapat pada sistem komputer dan menyediakan sekumpulan layanan (*system calls*) ke pemakai sehingga memudahkan dan membuat pengguna nyaman dalam pemanfaatan *resource* sistem komputer (Kelompok Kerja 21–28 IKI-20230, 2003).

Sistem komputer pada dasarnya terdiri dari empat komponen utama, yaitu perangkat-keras, sistem-operasi, program aplikasi dan para pengguna. Sistem

operasi berfungsi untuk mengatur dan mengawasi penggunaan perangkat keras oleh berbagai program aplikasi serta para pengguna. Sistem operasi mengatur penggunaan *resource* antar pengguna untuk menghindari konflik yang terjadi pada saat pengguna menggunakan *resource* yang sama, sehingga sistem operasi sering juga disebut *resource allocator*. Satu lagi fungsi penting sistem operasi ialah sebagai program pengendali yang bertujuan untuk menghindari kekeliruan (*error*) dan penggunaan *resource* yang tidak perlu.

2.2.1 Struktur Sistem Operasi

Pada kenyataannya tidak semua sistem operasi mempunyai struktur yang sama. Namun menurut Avi Silberschatz, Peter Galvin, dan Greg Gagne, umumnya sebuah sistem operasi modern mempunyai komponen (Kelompok Kerja 21–28 IKI-20230, 2003) sebagai berikut:

2.2.1.1 Managemen Proses

Proses adalah keadaan ketika sebuah program sedang di eksekusi. Sebuah proses membutuhkan beberapa sumber daya untuk menyelesaikan tugasnya. sumber daya tersebut dapat berupa *CPU time*, memori, berkas-berkas, dan perangkat-perangkat I/O. Sistem operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan managemen proses seperti:

1. Pembuatan, penghapusan proses pengguna dan sistem proses.
2. Menunda atau melanjutkan proses.
3. Menyediakan mekanisme untuk proses sinkronisasi.
4. Menyediakan mekanisme untuk proses komunikasi.

2.2.1.2 Managemen Memori Utama

Memori utama atau lebih dikenal sebagai memori adalah sebuah *array* yang besar dari *word* atau *byte*, yang ukurannya mencapai ratusan, ribuan, atau bahkan jutaan. Setiap *word* atau *byte* mempunyai alamat tersendiri. Memori Utama berfungsi sebagai tempat penyimpanan yang akses datanya digunakan oleh CPU atau perangkat I/O. Memori utama termasuk tempat penyimpanan data yang sementara (*volatile*), artinya data dapat hilang begitu sistem dimatikan. Sistem

operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan managemen memori seperti:

1. Menjaga *track* dari memori yang sedang digunakan dan siapa yang menggunakannya.
2. Memilih program yang akan di-*load* ke memori.
3. Mengalokasikan dan meng-dealokasikan ruang memori sesuai kebutuhan.

2.2.1.3 Managemen *Secondary-Storage*

Data yang disimpan dalam memori utama bersifat sementara dan jumlahnya sangat kecil. Oleh karena itu, untuk meyimpan keseluruhan data dan program komputer dibutuhkan *secondary-storage* yang bersifat permanen dan mampu menampung banyak data. Contoh dari *secondary-storage* adalah *harddisk*, *disket, flash disk* dan *zip drive*. Sistem operasi bertanggung-jawab atas aktivitas-aktivitas yang berkaitan dengan *disk-management* seperti: *free-space management*, alokasi penyimpanan, penjadualan disk.

2.2.1.4 Managemen Sistem I/O

Sering disebut *device manager*. Menyediakan "*device driver*" yang umum sehingga operasi I/O dapat seragam (membuka, membaca, menulis, menutup). Contoh: pengguna menggunakan operasi yang sama untuk membaca berkas pada *hard-disk,* CD-ROM dan *floppy disk*. Komponen Sistem Operasi untuk sistem I/O:

1. *Buffer*: menampung sementara data dari/ ke perangkat I/O.
2. *Spooling*: melakukan penjadualan pemakaian I/O sistem supaya lebih efisien (antrian dsb.).
3. Menyediakan *driver* untuk dapat melakukan operasi "rinci" untuk perangkat keras I/O tertentu.

2.2.1.5 Managemen Berkas

Berkas adalah kumpulan informasi yang berhubungan sesuai dengan tujuan pembuat berkas tersebut. Berkas dapat mempunyai struktur yang bersifat hirarkis (direktori, volume, dll.). Sistem operasi bertanggung-jawab:

1. Pembuatan dan penghapusan berkas.
2. Pembuatan dan penghapusan direktori.
3. Mendukung manipulasi berkas dan direktori.
4. Memetakan berkas ke *secondary storage*.
5. Mem-*backup* berkas ke media penyimpanan yang permanen (*non-volatile*).

2.2.1.6 Sistem Proteksi

Proteksi mengacu pada mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor, atau pengguna ke sistem sumber daya. Mekanisme proteksi harus:

1. membedakan antara penggunaan yang sudah diberi izin dan yang belum.
2. *specify the controls to be imposed.*
3. *provide a means of enforcement*.

2.2.1.7 Jaringan

Sistem terdistribusi adalah sekumpulan prosesor yang tidak berbagi memori atau *clock*. Tiap prosesor mempunyai memori sendiri. Prosesor-prosesor tersebut terhubung melalui jaringan komunikasi. Sistem terdistribusi menyediakan akses pengguna ke bermacam sumber-daya sistem. Akses tersebut menyebabkan:

1. *Computation speed-up.*
2. *Increased data availability.*
3. *Enhanced reliability*.

2.2.1.8 Command-Interpreter System

Sistem Operasi menunggu instruksi dari pengguna (*command driven*). Program yang membaca instruksi dan mengartikan *control statements* umumnya

disebut: *control-card interpreter*, *command-line interpreter,* dan *UNIX shell*. *Command-Interpreter System* sangat bervariasi dari satu sistem operasi kesistem operasi yang lain dan disesuaikan dengan tujuan dan teknologi *I/O devices* yang ada. Contohnya: *CLI, Windows, Pen-based (touch),* dan lain-lain.

2.3 Linux

Linux sangat mirip dengan sistem-sistem *UNIX*, hal ini dikarenakan kompatibilitas dengan *UNIX* merupakan tujuan utama desain dari proyek *Linux*. Perkembangan *Linux* dimulai pada tahun 1991, ketika mahasiswa *Finlandia* bernama *Linus Torvalds* menulis *Linux*, sebuah *kernel* untuk *prosesor* 80386, *prosesor 32-bit* pertama dalam kumpulan *CPU intel* yang cocok untuk PC. Pada awal perkembangannya, *source code Linux* disediakan secara bebas melalui *Internet*. Hasilnya, sejarah *Linux* merupakan kolaborasi banyak *programer* dari seluruh dunia, semuanya dilakukan secara eksklusif melalui *Internet*. Dari *kernel* awal yang hanya mengimplementasikan *subset* kecil dari sistem *UNIX*, sistem *Linux* telah bertumbuh dimana sudah mampu memasukkan banyak fungsi *UNIX*. *Kernel Linux* perlu dibedakan dari sebuah sistem *Linux*. *Kernel Linux* merupakan sebuah perangkat lunak orisinil yang dibuat oleh komunitas *Linux* sedangkan sistem *Linux*, yang diketahui saat ini mengandung banyak komponen yang dibuat sendiri atau dipinjam dari proyek lain.

2.3.1 Prinsip Rancangan Linux

Dalam rancangan keseluruhan, Linux menyerupai implementasi *UNIX nonmicro kernel* yang lain. Ia adalah sistem yang *multiuser*, *multitasking* dengan seperangkat lengkap alat-alat yang kompatibel dengan *UNIX*. Sistem berkas *Linux* mengikuti *semantik* tradisional *UNIX*, dan model jaringan standar *UNIX* diimplementasikan secara keseluruhan. Ciri internal rancangan *Linux* telah dipengaruhi oleh sejarah perkembangan sistem operasi ini. Walaupun *Linux* dapat berjalan pada berbagai macam *platform*, pada awalnya *Linux* dikembangkan secara eksklusif pada arsitektur PC. Sebagian besar dari pengembangan awal tersebut dilakukan oleh peminat individual, bukan oleh fasilitas riset yang

memiliki dana besar, sehingga dari awal *Linux* berusaha untuk memasukkan fungsionalitas sebanyak mungkin dengan dana yang sangat terbatas. Saat ini, *Linux* dapat berjalan baik pada mesin *multiprocessor* dengan *main memory* yang sangat besar dan ukuran *disk space* yang juga sangat besar, namun tetap mampu beroperasi dengan baik dengan jumlah *RAM* yang lebih kecil. Akibat dari semakin berkembangnya teknologi PC, kernel Linux juga semakin lengkap dalam mengimplementasikan fungsi *UNIX*. Tujuan utama perancangan *Linux* adalah cepat dan efisien, tetapi akhir-akhir ini konsentrasi perkembangan *Linux* lebih pada tujuan rancangan yang ketiga yaitu standarisasi (Bouet, 2002). Standar *POSIX* (*Portable Operating System Interface based on uniX*) terdiri dari kumpulan spesifikasi dari beberapa aspek yang berbeda tingkah lakunya terhadap sistem operasi lain. Dokumen *POSIX* berfungsi sebagai sistem operasi biasa dan untuk ekstensi seperti proses untuk thread dan operasi *real-time*. Linux dirancang agar sesuai dengan dokumen *POSIX* yang relevan. Sedikitnya ada dua distribusi Linux yang sudah memperoleh sertifikasi ofisial *POSIX*.

Karena Linux memberikan antarmuka standar ke *programer* dan pengguna, *Linux* tidak membuat banyak kejutan kepada siapa pun yang sudah terbiasa dengan *UNIX*. Namun *interface* pemrograman *Linux* merujuk pada semantik *SVR4 UNIX* daripada *BSD*. Kumpulan pustaka yang berbeda tersedia untuk mengimplementasi *semantik BSD* di tempat dimana kedua kelakuan sangat berbeda. Ada banyak standar lain di dunia *UNIX*, tetapi sertifikasi penuh dari Linux terhadap standar lain *UNIX* terkadang menjadi lambat karena lebih sering tersedia dengan harga tertentu (tidak secara bebas), dan ada harga yang harus dibayar jika melibatkan sertifikasi persetujuan atau kecocokan sebuah sistem operasi terhadap kebanyakan standar. Bagaimana pun juga mendukung aplikasi yang luas adalah penting untuk suatu sistem operasi, sehingga sehingga standar implementasi merupakan tujuan utama pengembangan *Linux*, walaupun implementasinya tidak sah secara formal. Selain standar *POSIX*, *Linux* saat ini mendukung ekstensi *thread POSIX* dan *subset* dari *ekstensi* untuk kontrol proses *real-time POSIX*.

Sistem *Linux* terdiri dari tiga bagian kode penting:

1. *Kernel***.** Bertanggung-jawab memelihara semua abstraksi penting dari sistem operasi, termasuk hal-hal seperti memori *virtual* dan proses-proses.
2. Pustaka sistem (*System library*). Menentukan kumpulan fungsi standar dimana aplikasi dapat berinteraksi dengan *kernel*, dan mengimplementasi hampir semua fungsi sistem operasi yang tidak memerlukan hak penuh atas *kernel*.
3. Utilitas sistem dan *Service*. Program yang melakukan pekerjaan manajemen secara individual.

2.3.1.1 Kernel

Walaupun berbagai sistem operasi modern telah mengadopsi suatu arsitektur *message-passing* untuk *kernel* internal mereka, *Linux* tetap memakai model historis *UNIX*. *Kernel* diciptakan sebagai biner yang tunggal dan monolitis. Alasan utamanya adalah untuk meningkatkan kinerja, karena semua struktur data dan kode kernel disimpan dalam satu *address space*, alih konteks tidak diperlukan ketika sebuah proses memanggil sebuah fungsi sistem operasi atau ketika interupsi perangkat keras dikirim. Tidak hanya penjadwalan inti dan kode memori *virtual* yang menempati *address space* ini, tetapi juga semua kode *kernel*, termasuk semua *device drivers*, sistem berkas, dan kode jaringan, hadir dalam satu *address space* yang sama (Kroah, 2005).

Kernel Linux membentuk inti dari sistem operasi *Linux*. Dia menyediakan semua fungsi yang diperlukan untuk menjalankan proses, dan menyediakan layanan sistem untuk memberikan pengaturan dan proteksi akses ke sumber daya perangkat keras. *Kernel* mengimplementasi semua fitur yang diperlukan supaya dapat bekerja sebagai sistem operasi. Namun, jika sendiri, sistem operasi yang disediakan oleh *kernel Linux* sama sekali tidak mirip dengan sistem *UNIX*. Dia tidak memiliki banyak fitur ekstra *UNIX*, dan fitur yang disediakan tidak selalu dalam format yang diharapkan oleh aplikasi *UNIX*. *Interface* dari sistem operasi yang terlihat oleh aplikasi yang sedang berjalan tidak ditangani langsung oleh

kernel, akan tetapi aplikasi membuat panggilan (*calls*) ke perpustakaan sistem, yang kemudian memanggil layanan sistem operasi yang dibutuhkan.

Secara umum *kernel* yang dikompilasi terbagi menjadi dua bagian, yaitu:

1. *Kernel modular*

   Salah satu keuntungan kernel yang bersifat *modular*, pergantian *hardware* menjadi lebih mudah, karena *probing hardware* dari suatu modul dapat mendeteksi adanya *hardware* baru atau jika belum tersedia sistem dapat mem-*build* satu *modules* saja. Kerugiannya adalah pada *kernel* modular seluruh *device drivers* yang tersedia ter-*include* ke dalam kernel sehingga beberapa *devices drivers* yang sebenarnya tidak kita butuhkan juga turut ter-*load* sewaktu *kernel* dieksekusi dan pada *modules* yang ada pada *kernel* modular relatif rentan terhadap masalah *security*, karena biasanya ada beberapa *script kiddies* yang memasukkan suatu modul ke dalam kernel (dengan harapan proses yang dimilikinya tidak diketahui oleh *admin* sistem yang bersangkutan)

2. *Kernel monolitik*

   Dari segi *security*, sebuah *kernel monolitik* akan relatif aman. Namun dari segi kemudahan, jika ada penambahan atau pergantian suatu *hardware*, maka otomatis anda harus mengkompilasi ulang *kernel* dan harus menyertakan *device driver* yang sesuai dengan *hardware* yang akan di-*install* ke dalam *PC router*. Selain itu hanya *devices driver* yang kita sertakan pada saat konfigurasi *kernel* saja yang akan dimuat ke dalam *kernel*, sehingga menghemat proses *booting* dan ukuran *kernel*.

2.3.1.2 Pustaka Sistem (*System Library*)

Pustaka sistem menyediakan berbagai tipe fungsi. Pada *level* yang paling sederhana, mereka membolehkan aplikasi melakukan permintaan pada layanan sistem *kernel*. Membuat suatu *system call* melibatkan transfer kontrol dari mode pengguna yang tidak penting ke mode kernel yang penting, rincian dari transfer ini berbeda pada masing-masing arsitektur. Pustaka bertugas untuk

mengumpulkan argumen *system-call* dan, jika perlu, mengatur argumen tersebut dalam bentuk khusus yang diperlukan untuk melakukan *system call*.

Pustaka juga dapat menyediakan versi lebih kompleks dari *system call* dasar. Contohnya, fungsi *buffered file-handling* dari bahasa C semuanya diimplementasikan dalam pustaka sistem, yang memberikan kontrol lebih baik terhadap berkas M/K daripada *system call kernel* dasar. pustaka juga menyediakan rutin yang tidak ada hubungan dengan *system call*, seperti algoritma penyusunan (*sorting*), fungsi matematika, dan rutin manipulasi *string* (*string manipulation*). Semua fungsi yang diperlukan untuk mendukung jalannya aplikasi *UNIX* atau *POSIX* diimplementasikan dalam pustaka sistem.

2.3.1.3 Utilitas Sistem (*Service*)

Sistem *Linux* mengandung banyak program-program pengguna-mode, utilitas sistem dan utilitas pengguna. Utilitas sistem termasuk semua program yang diperlukan untuk menginisialisasi sistem, seperti program untuk konfigurasi alat jaringan (*network device*) atau untuk *load modul kernel*. Program *server* yang berjalan secara kontinu juga termasuk sebagai utilitas sistem. Program semacam ini mengatur permintaan pengguna *login*, koneksi jaringan yang masuk, dan antrian *printer*. Tidak semua utilitas standar melakukan fungsi administrasi sistem yang penting. Lingkungan pengguna *UNIX* mengandung utilitas standar dalam jumlah besar untuk melakukan pekerjaan sehari-hari, seperti membuat daftar direktori, memindahkan dan menghapus file, atau menunjukkan isi dari sebuah file. Utilitas yang lebih kompleks dapat melakukan fungsi *text-processing*, seperti menyusun data tekstual atau melakukan pattern searches pada input teks. Jika digabung, utilitas-utilitas tersebut membentuk kumpulan alat standar yang diharapkan oleh pengguna pada sistem UNIX mana saja; walaupun tidak melakukan fungsi sistem operasi apa pun, utilitas tetap merupakan bagian penting dari sistem *Linux* dasar.

2.3.2 PC *Router* Linux

PC *router* merupakan suatu perangkat komputer yang memiliki suatu sistem operasi yang memiliki lebih dari satu *network interface card* dan dipergunakan untuk menghubungkan beberapa *network* yang berlainan (Odom, 2004). Fungsi *PC router* sama dengan *dedicated router*, namun dari sisi performa dan kinerja, tidak dapat dipungkiri *dedicated router* lebih unggul dalam beberapa hal. PC *router* dapat di-*install* oleh sistem operasi apa saja, asalkan sistem operasi itu mendukung untuk *routing*. Ada beberapa hal yang mempengaruhi kinerja dari sebuah *router* (Lammle, 2006), yaitu:

1. *Routing* protokol yang dipakai sangat berdampak pada performa *router*, karena dalam mencari dan mengupdate *routing table*, *router* akan menggunakan alokasi *processor* dan *memory* yang besar.
2. *Bandwith* yang dialirkan oleh *traffic* yang sibuk juga menjadi hal yang mempengaruhi kinerja *router*, *traffic* yang terlalu sibuk dapat menyebabkan kongesti pada *network*.
3. *Interface* yang digunakan pada *router* juga berdampak pada kinerja *router*. Jika *interface* yang dialiri oleh *traffic* besar namun kapasitas untuk mengaliri *traffic* data juga besar, kongesti pada *network* dapat dihindari.

2.3.3 Distro Linux

Distro Linux (singkatan dari distribusi *Linux*) adalah sebutan untuk sistem operasi komputer mirip *Unix* yang menggunakan *kernel Linux*. Distribusi *Linux* bisa berupa perangkat lunak bebas dan bisa juga berupa perangkat lunak komersial seperti *Red Hat Enterprise*, *SuSE*, dan lain-lain. Ada banyak distribusi atau distro *Linux* yang telah muncul. Beberapa bertahan dan berkembang, bahkan sampai menghasilkan distro turunan, contohnya adalah:

1. *Debian GNU/Linux* **(*http://www.debian.org/*).** *Debian GNU/Linux* adalah distro *non*-komersial yang dihasilkan oleh para sukarelawan dari seluruh dunia yang saling bekerjasama melalui *Internet*. Distro ini

menginginkan adanya semangat *open-source* yang harus tetap ada pada *Debian*. Kedinamisan distro ini membuat setiap rilis paket-paketnya di-*update* setiap waktu dan dapat diakses melalui utilitas *apt-get*. *Apt-get* adalah sebuah utilitas baris-perintah yang dapat digunakan secara dinamis untuk meng-*upgrade* sistem *Debian GNU/Linux* melalui *apt-repository* jaringan *archive Debian* yang luas. *Milis* dan forum *debian* selalu penuh dengan pesan-pesan baik mengenai *bug*, masalah, *sharing*, dan lain-lain. Dengan adanya sistem komunikasi ini *bug* dan masalah keamanan pada tiap paket dapat dilaporkan oleh para pengguna dan pengembang *Debian* dengan cepat. Keuntungan dari *Debian* adalah *upgradability*, ketergantungan antar paket didefinisikan dengan baik, dan pengembangannya secara terbuka. Beberapa proyek dan turunan *Debian GNU/Linux*:

a. *De2*, *http://de2.vlsm.org/*

b. *Knoppix*, *http://www.knoppix.org/*

c. *Debian JP*, *http://www.debian.linux.or.jp/*

d. *Ubuntu*, *http://www.ubuntu.org/*

2. *Red Hat Linux* **(*http://www.redhat.com/*).** *Red Hat* adalah distro yang cukup populer di kalangan pengembang dan perusahaan *Linux*. Dukungan-dukungan secara teknis, pelatihan, sertifikasi, aplikasi pengembangan, dan bergabungnya para *hacker kernel* dan *free-software* seperti Alan Cox, Michael Johnson, Stephen Tweedie menjadikan Red Hat berkembang cepat dan digunakan pada perusahaan. Poin terbesar dari distro ini adalah *Red Hat Package Manager* (RPM). RPM adalah sebuah perangkat lunak untuk memanajemen paket-paket pada sistem *Linux* kita dan dianggap sebagai standar *de-facto* dalam pemaketan pada distro-distro turunannya dan yang mendukung distro ini secara luas.

3. *Slackware* **(*http://www.slackware.com/*).** Distronya Patrick Volkerding yang terkenal pertama kali setelah SLS. *Slackware* dikenal lebih dekat dengan gaya *UNIX*, sederhana, stabil, mudah di-*custom*, dan didesain untuk komputer 386/486 atau lebih tinggi. Distro ini termasuk distro yang *cryptic* dan *manual* sekali bagi pemula *Linux*, tapi dengan menggunakan distro ini beberapa penggunanya dapat mengetahui banyak cara kerja sistem dan distro tersebut. *Debian* adalah salah satu distro selain *Slackware* yang masuk dalam kategori ini. Sebagian besar aktivitas konfigurasi di Slackware dilakukan secara *manual* (tidak ada *tool* seperti *Yast* pada S.U.S.E ataupun *Linuxconf* pada *RedHat*).

4. S.u.S.E. **(*http://www.suse.com/*).** S.u.S.E. adalah distro yang populer di Jerman dan Eropa, terkenal akan dukungan *driver* VGA-nya dan *YasT*. S.u.S.E tersedia secara komersial dan untuk versi *GPL*-nya dapat diinstal melalui *ftp* di situs S.u.S.E. Instalasi berbasis menu grafis dari CD-ROM, *disket boot modular*, 400-halaman buku referensi, dukungan teknis, dukungan *driver-driver* terutama VGA dan *tool* administrasi sistem S.u.S.E., *YaST*, membuat beberapa pengguna memilih distro ini. S.u.S.E. juga terlibat dalam pembuatan X *server* (*video driver*) untuk proyek *XFree86* sehingga X *server* distro ini mendukung kartu grafis baru. S.U.S.E. menggunakan dua sistem pemaketan yaitu RPM (versi lama) dan SPM, S.U.S.E. *Package Manager* (versi baru).

5. Turbo Linux **(*http://www.turbolinux.com/*).** *TurboLinux* menargetkan pada produk berbasis *Linux* dengan kinerja tinggi untuk pasar *workstation* dan *server* terutama untuk penggunaan *clustering* dan orientasinya ke perusahaan. Beberapa produk-produknya: *TurboLinux Workstation* untuk dekstopnya, *TurboLinux Server* untuk *backend server* dengan kinerja tinggi terutama untuk penggunaan bisnis di perusahaan, *e-commerce* dan transaksi B2B (*Business-to-Business*). Salah satu produknya *TurboCluster Server* ditargetkan untuk pembuatan *server cluster* yang berskala luas dan dapat digunakan 25 cluster node atau

lebih. *TurboCluster server* ini pernah memenangkan poling *Best Web Solution* dari *editor Linux Journal*. *enFuzion*, satu lagi produk yang berbasis pada konsep sederhana dan *powerful* yang dinamakan '*parametric execution*'. *enFuzion* akan merubah jaringan komputer perusahaan menjadi *super computer* dengan kecepatan tinggi dan '*fault tolerant*'. Pengguna produk dan layanan *TurboLinux* terbanyak adalah perusahaan dan perorangan di Jepang dan Asia.

Untuk mendapatkan distro *linux,* anda dapat men-*download*-nya langsung dari situs distributor distro bersangkutan, atau membelinya dari penjual lokal.

2.3.4 Distro Linux LiveCD *Router*

Linux LiveCD Router adalah distribusi *Linux* khusus didisain untuk berbagi koneksi *Internet* nirkabel pita lebar *WiFi (share a broadband connection over WiFi)*. Dapat digunakan untuk koneksi via DSL, *cable modem*, T1, ISDN, dan *dial-up*. Linux LiveCD *Router* dapat digunakan juga sebagai *firewall*, atau sebagai *access point* menggunakan *WiFi cards*. Pada penggunaannya *Linux LiveCD Router* tidak perlu di instalasi, cukup menyediakan sebuah PC dengan pemutar *CDRom* untuk *boot* dan menjalankan dengan *PocketCD* yang praktis. *Linux LiveCD Router* menggunakan distro *Linux Slackware* sebagai *main system* dan dapat didistribusikan secara gratis. Namun untuk dukungan *WiFi* dan dukungan *bandwith Linux LiveCD Router* membatasi limit pada mode *free user*.

2.3.5 GNU General Public License

Hampir semua lisensi dari perangkat lunak dirancang untuk merebut kebebasan anda dan mengubahnya. Sebaliknya, Lisensi Publik Umum GNU (GNU *General Public License*) bertujuan untuk menjamin kebebasan anda untuk berbagi dan mengubah perangkat lunak bebas, untuk menjamin bahwa perangkat lunak tersebut tetap bebas bagi penggunanya. *General Public License* ini dapat diberlakukan terhadap hampir semua perangkat lunak *Free Software Foundation* dan program lain apa pun yang penciptanya mau menggunakan Lisensi ini. (Beberapa perangkat lunak *Free Software Foundation* lainnya menggunakan

GNU *Library Public License*.) Anda dapat memberlakukannya terhadap program Anda juga.

Ketika kita berbicara tentang perangkat lunak bebas, kita mengacu kepada kebebasan, bukan harga. Lisensi Publik Umum kami dirancang untuk menjamin bahwa Anda memiliki kebebasan untuk mendistribusikan salinan dari perangkat lunak bebas (dan memberi harga untuk jasa tersebut jika Anda mau), mendapatkan *source code* atau bisa mendapatkannya jika Anda mau, mengubah suatu perangkat lunak atau menggunakan bagian dari perangkat lunak tersebut dalam suatu program baru yang juga bebas dan mengetahui bahwa anda dapat melakukan semua hal ini.

Untuk melindungi hak-hak Anda, kami perlu membuat batasan-batasan yang melarang orang lain untuk dapat menolak hak-hak Anda atau membuat Anda menyerahkan hak-hak Anda tersebut. Batasan-batasan ini diterjemahkan menjadi beberapa tanggung jawab bagi Anda jika Anda mendistribusikan salinan dari suatu perangkat lunak, atau memodifikasinya. Sebagai contoh, jika Anda mendistribusikan salinan dari suatu program, baik secara gratis atau dengan biaya, Anda harus memberi semua hak-hak Anda kepada si penerima. Anda juga harus menjamin bahwa si penerima tersebut mendapatkan atau bisa mendapatkan *source code*-nya. Lindungi hak-hak Anda dengan dua langkah:

1. Hak cipta terhadap perangkat lunak tersebut, dan
2. Tawarkan Lisensi ini kepada *user* dengan izin legal untuk menyalin, mendistribusikan dan/atau memodifikasi perangkat lunak tersebut.

Demi perlindungan bagi si pencipta dan *user* lainnya, pastikan bahwa semua orang mengerti bahwa tidak ada garansi bagi perangkat lunak bebas. Jika perangkat lunak tersebut dimodifikasi oleh orang lain dan didistribusikan, *user* hendaknya mengetahui bahwa apa yang mereka punyai bukanlah perangkat lunak yang aslinya, sehingga masalah apa pun yang ditimbulkan oleh orang lain tidak mencerminkan reputasi pencipta perangkat lunak yang asli.

Terakhir, *program* bebas apa pun terancam terus menerus oleh hak *paten* perangkat lunak. Sebaiknya hindari bahaya yang memungkinkan *redistributor program* yang bebas bisa mendapatkan hak paten untuk dirinya sendiri, yang

mengakibatkan program tersebut menjadi tak bebas. Untuk mencegah hal ini, nyatakan dengan jelas bahwa hak paten apa pun harus dilisensikan bagi semua orang, atau tidak sama sekali.

BAB III

ANALISA DAN PERANCANGAN

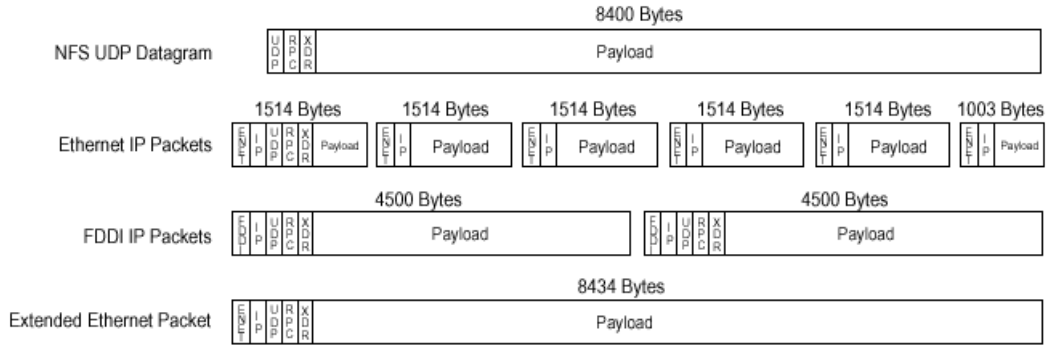
3.1 Analisa Fungsi *Kernel* dan *Services* pada *Linux*

Dalam perancangan *Linux* terdapat tiga hal yang utama yaitu *kernel*, *system library* dan *services*. *Kernel* merupakan inti dari sistem operasi *Linux* yang dalam hal ini menyediakan segala fitur yang dibutuhkan dalam sebuah sistem operasi, seperti konektivitas antar *devices*, manajemen *drivers* perangkat, serta manajemen struktur I/O antar perangkat sehingga tidak menimbulkan suatu konflik antar perangkat. Didalam *kernel* terdiri dari *device driver* dan modul-modul *kernel* yang dapat dikompilasi, dihapus, dan dipanggil secara terpisah dari bagian *kernel* lainnya saat dibutuhkan. Modul *kernel* dapat menambah fungsionalitas *kernel* tanpa perlu me-*reboot* sistem. Secara teori tidak ada yang dapat membatasi apa yang dapat dilakukan oleh modul *kernel*. *Kernel* modul dapat mengimplementasikan antara lain sistem berkas, protokol jaringan dan *device driver*. Secara default ada beberapa *device driver* yang langsung termuat kedalam kernel, namun tidak semua *devices driver* yang ada dimuat kedalam *memory*. Sistem harus melakukan inisialisasi (*probing*) terhadap perangkat, yang kemudian mencari *devices driver* yang sesuai dengan perangkat dan kemudian akan di-*load* kedalam *memory* untuk dipergunakan oleh perangkat. *Kernel* dapat dibuat optimal dengan mengkompilasi *kernel*, karena beberapa modul dan *device driver* yang tidak diperlukan dan akan ter-*load* ke *memory* yang akan memakan *space* dari *main memory*. Hal inilah yang banyak dijadikan alasan bahwa *kernel* tersebut menjadi tidak efisien terhadap sistem. Begitu juga dengan *service*. *Service* pada bawaan *Linux* juga merupakan penderitaan bagi *managemen memory* pada sistem. Ada beberapa *service* yang ter-*load* dan mungkin tidak dibutuhkan, tergantung kebutuhan personal dan fungsi dari *service* itu sendiri. Bagi *PC router* dengan perangkat baru dan alokasi *memory* yang besar mungkin tidak menjadi

kendala. Namun bagi *PC router* dengan alokasi *memory* yang sedikit dengan perangkat lama, tentunya akan menjadi beban bagi sistem.

3.2 Pengaruh MTU Terhadap Transfer Data

MTU (*Maximum Transfer Unit*) adalah standar frame atau panjang maksimum data yang dapat ditransfer pada *Ethernet* dan *Fast Ethernet* yang berkisar antara 40 bytes sampai 1500 *bytes* (*plus header* dan *tailer*). Besar MTU mempengaruhi jumlah paket yang akan ditransfer kedalam sebuah jaringan. Apabila suatu transfer paket yang sangat besar dilakukan dengan nilai MTU yang kecil maka jumlah paket yang dikirimkan akan menjadi lebih banyak bila dibandingkan dengan pengiriman paket yang memiliki MTU yang besar. Selain itu setiap paket (*network layer OSI*) yang berasal dari layer atas (*layer transport*) akan ditambahkan *header* sebelum ditangani pada layer berikutnya (*data link layer*) sehingga menyebabkan terjadinya *over header* yang akan menambah beban pada trafik. (Global Knowledge. 1999)



Gambar 3.1 Pemborosan jumlah paket dan over header

Ada beberapa pengaruh dari besarnya nilai MTU terhadap jumlah paket dan *over header* dalam performa jaringan baik itu dari performa *transfer* data, pemprosesan data maupun pengaruh terhadap efektifitas trafik. Secara lebih rinci besar nilai MTU dapat mempengaruhi hal-hal sebagai berikut:

1. *Throughput*
2. *Over header*
3. *Effective Bandwidth*

3.3 Perancangan Simulasi

Dalam tahap perancangan ini penulis akan menjelaskan konfigurasi dari rancangan simulasi yang meliputi kebutuhan perangkat keras dan perangkat lunak, perancangan konfigurasi jaringan dan cara menggunakannya serta penjelasan mengenai aplikasi-aplikasi pendukung untuk melakukan simulasi, monitoring dan pengumpulan data statistik dari hasil simulasi. Pada bagian aplikasi yang digunakan untuk simulasi akan dijelaskan secara singkat fungsi dan tujuan pemakaian aplikasi tersebut. Sedangkan penerapan lebih teknis dari perancangan ini akan dijelaskan pada tahap simulasi dan hasil.

3.3.1 Kebutuhan Perangkat Keras (*Hardware*)

Perangkat keras yang dibutuhkan adalah;

1. *PC Router* dengan spesifikasi:
   a. CPU : *Pentium* 233 MMX
   b. Memory : 96MB
   c. Hardisk : 20 GB
   d. CD-Rom : *Samsung* 52x
   e. NIC : *Intel Pro 100 PCI* dan *Intel 8255x PCI*
2. Dua buah PC Client dengan spesifikasi:
   a. CPU : *Pentium 3*
   b. Memory : 128MB
   c. Hardisk : 40 GB
   d. NIC : *Intel 8255x PCI / Intel Pro 100 PCI*
3. UTP *Cable Category 5*
4. *Switch 8 Port*

3.3.2 Kebutuhan Perangkat Lunak (*Software*)

Perangkat lunak yang dibutuhkan meliputi dua hal, yaitu *operating system* dan aplikasi. *Operating system* yang digunakan di *PC router* dan *client* memiliki perbedaan. *PC router* di-*install* dengan *operating system Red Hat 9* dengan *kernel*

bawaan 2.4.20-8, sedangkan kedua client menggunakan *Windows XP Service Pack 2*. Sedangkan aplikasi yang dibutuhkan sebagai alat simulasi adalah:

1. *Ftp Server*

   FTP server yang digunakan adalah *3Cdaemon*. *3CDaemon* merupakan sebuah *FTP* dan *TFTP server*, sekaligus merangkap sebagai *Syslog Server*. *FTP server* ini akan di-*install* pada salah satu *client*. Sehingga *FTP Client* nantinya dapat terhubungan dengan *FTP Server*.

2. *FTP Client*

   FTP client yang digunakan adalah *FTP client default* bawaan yang ada pada *windows*.

3. *Network Performance Measurement*

   Alat pengukur performa jaringan yang digunakan adalah:

   a. *Iperf - 1.6.3 - win32* yaitu aplikasi *freeware* yang digunakan untuk mengukur dan mengkonfigurasi pengiriman paket data baik yang menggunakan *TCP* maupun yang menggunakan *UDP*. Namun *Iperf* pada pengujian hanya menggunakan protokol TCP. Fungsi-fungsi yang terdapat dalam aplikasi *Iperf* antara lain:

      1. TCP
         a. Memantau kapasitas *bandwidth*.
         b. *Report MSS/MTU size*.
         c. Mendukung TCP *window size via socket buffers*.
         d. Melihat hasil *throughput* data yang dikirim
      2. UDP
         a. Dapat membuat *UDP stream* dari *bandwidth* yang spesifik.
         b. Memonitoring jumlah *packet loss*.
         c. Memonitoring *delay jitter*.

   b. Alat pengukur performa *network* yang selanjutnya dipakai adalah *Qcheck* dari IXIA. *Qcheck* dapat mengukur performa dari paket TCP dan UDP. Qcheck tidak sebatas mengukur performa *network* melalui *response time*, tetapi juga mengukur *throughput* serta *packet loss* yang ada dan juga *streaming* antar dua buah *end point*.

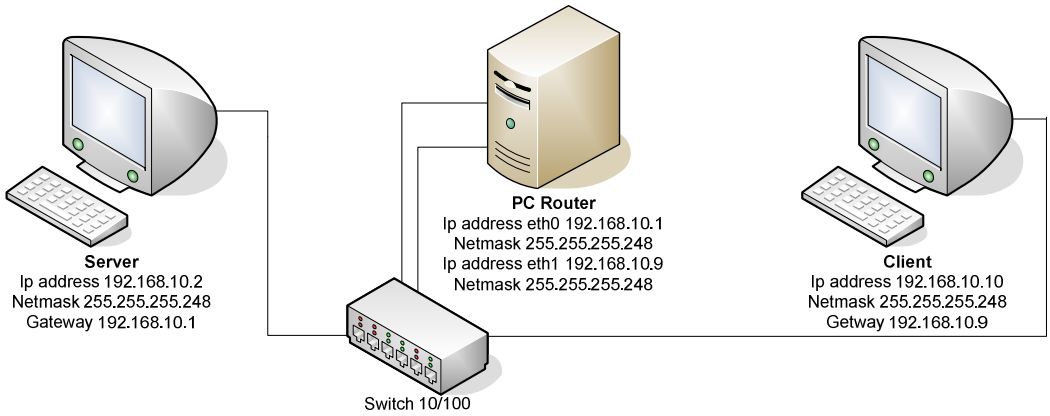
4. *SSH client*

   SSH client digunakan untuk membentuk koneksi dengan *remote SSH server* yang ada pada *PC router*, sehingga *router* dapat dikonfigurasi dengan mudah dari *console* yang ada di-*client*. *SSH client* yang digunakan adalah *SecureCRT* 4.0.

3.3.3 Perancangan Struktur Jaringan

Struktur rancangan jaringan yang akan digunakan terdiri dari dua unit PC client dan satu unit PC router dengan rancangan konfigurasi sebagai berikut:

1. Terdiri dari dua buah client yang terhubung langsung ke PC router menggunakan kabel UTP Category 5.
2. IP *address* untuk masing-masing *client* dan *router* akan ditentukan. Karena IP *address* yang digunakan tidak terlalu banyak, maka dapat dilakukan *subnetting*. *Subnetting* akan dilakukan untuk IP *address* kelas C, yaitu pada alamat *network* 192.168.10.0 *mask* 255.255.255.248
3. Dari *subnetting* yang dilakukan, didapat data sebagai berikut:
   a. IP *address* yang *valid* dalam satu *network* berjumlah 6 buah.
   b. Alamat *network* pertama adalah 192.168.10.0 dengan alamat *broadcast* 192.168.10.7 sedangkan untuk alamat *network* kedua didapat *network address* 192.168.10.8 dengan alamat *broadcast* 192.168.10.15
   c. *Range* IP *address valid* yang dapat digunakan adalah 192.168.10.1 – 192.168.10.6 dan 192.168.10.9 – 192.168.10.14
   d. Dan alamat setiap *subnet* dapat dicari dengan rumus { 256 – alamat mask } yaitu 256 – 248 = 8. Jadi alamat *valid* dari setiap subnet adalah 0, 8, 16, 24, 32, 40, 48, 64, … , 248. Jadi alamat *network*-nya dapat berupa 192.168.10.0, 192.168.10.8 atau kelipatannya.
   e. Dari data diatas dapat dibentuk sebuah gambar struktur *network* yang dibangun, seperti gambar dibawah ini.



Gambar 3.2 Perancangan Jaringan

3.3.4 Konfigurasi PC *Router*

Sebelum memulai koneksi antar *host*, *router* harus dikonfigurasi terlebih dahulu. Secara umum, mengkonfigurasi *router* berbasiskan *Linux* lebih banyak dilakukan dengan mengetikan baris perintah melalui *console*. Tidak seperti OS *Microsoft* yang banyak mengkonfigurasi melalui *desktop GUI* (*Graphic User Interface*) yang dapat di-*click* melalui tombol *mouse*. *Linux* yang merupakan turunan *Unix* lebih banyak mengkonfigurasi baris-baris perintah melalui CLI (*Command Line Interpreter*). PC *router Linux* juga tidak di-*install* GUI, bukan karena masalah *compatibility* atau karena *Linux* tidak mendukung GUI, namun lebih kepada efisiensi kinerja serta performa dari sebuah *router*..

3.3.4.1 Konfigurasi IP Address

Pada saat installasi *Linux*, sebenarnya ada permintaan untuk mengkonfigurasi *ip address* dari *interface* yang telah terdeteksi oleh *Linux*. Namun bila langkah ini telah terlewatkan atau mungkin terlupakan, dapat dikonfigurasi dengan mengetikan baris perintah *ifconfig*. Pada *interface0* yang akan diberikan *ip address* 192.168.10.1 *netmask* 255.255.255.248 dan *broadcast address* 192.168.10.7. Perintah tersebut dapat dengan: *Ifconfig eth0 192.168.10.1 netmask 255.255.255.248 broadcast 192.168.10.7.* Demikian pada *interface1* yang juga akan diberikan *ip address* 192.168.10.9 *netmask* 255.255.255.248 dan *broadcast address* 192.168.10.15. Perintah tersebut dapat diketikan dengan:

Ifconfig eth1 192.168.10.9 netmask 255.255.255.248 broadcast 192.168.10.15

Ada beberapa cara yang lebih mudah lagi, yaitu dengan mengedit file konfigurasi dari ethernet0 dan ethernet1 yang berada pada */etc/sysconfig/network-scripts/ifcfg-eth[number]*. Pada file konfigurasi tersebut dapat diisikan ip address yang tertera pada kolom dibawah ini.

/etc/sysconfig/network-scripts/ifcfg-eth0	*/etc/sysconfig/network-scripts/ifcfg-eth1*
DEVICE=eth0 ONBOOT=yes BOOTPROTO=static IPADDR=192.168.10.1 NETMASK=255.255.255.248 BROADCAST=192.168.10.7 NETWORK=192.168.10.0	DEVICE=eth1 ONBOOT=yes BOOTPROTO=static IPADDR=192.168.10.9 NETMASK=255.255.255.248 BROADCAST=192.168.10.15 NETWORK=192.168.10.8

Tabel 3.1 Tabel Konfigurasi Ethernet pada PC router.

3.3.4.2 Konfigurasi IP Routing

Setelah *ip address* pada *router* dikonfigurasi, tahap selanjutnya adalah mengkonfigurasi *ip routing*. *Ip routing* diperlukan oleh sebuah *router* karena fungsi *router* secara default adalah untuk mem-*forward* paket data dari suatu *network* ke *network* yang lain yang berbeda. Semenjak *kernel 2.4* keatas, *ip routing* secara *default* tidak diaktifkan lagi oleh *system (*. Untuk itu diperlukan untuk mengaktifkan *ip routing* / *ip fowarding* agar paket data dapat berjalan pada dua *network* yang berbeda.

Cara mengaktifkan *ip fowarding* pada *Linux* adalah dengan mengetikan baris perintah berikut pada console: *root # echo "1" > /proc/sys/net/ipv4/ip_forward*

Perintah diatas berguna untuk memberikan nilai 1 pada *script* file *ip_foward* sehingga mebuat proses *ip routng / ip fowarding* aktif. Dapat juga mengaktifkan *ip routing* pada saat *start up*, sehingga prosses *fowarding packet* dapat berjalan otomatis pada saat *router* dihidupkan. Untuk itu *edit* file */etc/sysctl.conf* menggunakan *editor* program *vi* yang populer dikalangan pengguna *Linux*. Pada baris *net.ipv4.ip_forward = 0* beri dengan nilai 1. Sehingga barisnya menjadi: *net.ipv4.ip_forward = 1*. Kemudian *save* dan *restart network service* dengan perintah:

```
[root@rh-router /]# service network restart
Shutting down interface eth0:  [  OK  ]
Shutting down interface eth1:  [  OK  ]
Shutting down loopback interface:  [  OK  ]
Disabling IPv4 packet forwarding:  [  OK  ]
Setting network parameters:  [  OK  ]
Bringing up loopback interface:  [  OK  ]
Bringing up interface eth0:  [  OK  ]
Bringing up interface eth1:  [  OK  ]
```

3.3.5 Koneksi ke SSH Server pada PC router

Setelah semua konfigurasi yang dibutuhkan untuk koneksi *TCP/IP* telah di lakukan, baik pada *router* maupun *client*. Maka *remote* dapat dilakukan terhadap *router* melalui PC *client* menggunakan *SSH client SecureCRT 4.0*. *SecureCRT* merupakan *console remote program* untuk mengetikan baris perintah. Yang digunakan untuk mengkonfigurasi suatu devices, baik itu berupa *router*, *firewall*, *switch*, maupun perangkat jaringan yang dapat di-*remote* dan mendukung koneksi melalui SSH (*Secure SHell*). Ada beberapa hal yang perlu diperhatikan dalam meng-koneksikan suatu perangkat melalui SSH ;

1. Pastikan SSH *server* telah diaktifkan.

   Setelah konfigurasi IP *address* pada *router* dan *client* telah di-*set* seperti gambar diatas, maka pastikan dahulu bahwa SSH *server* pada *router* telah aktif. Baris perintah dapat diketikan:

   ```
   [root@rh-router root]# service sshd status
   sshd (pid 2882 2826 1136) is running...
   ```

   apabila perintah yang diketikkan menghasilkan *status*

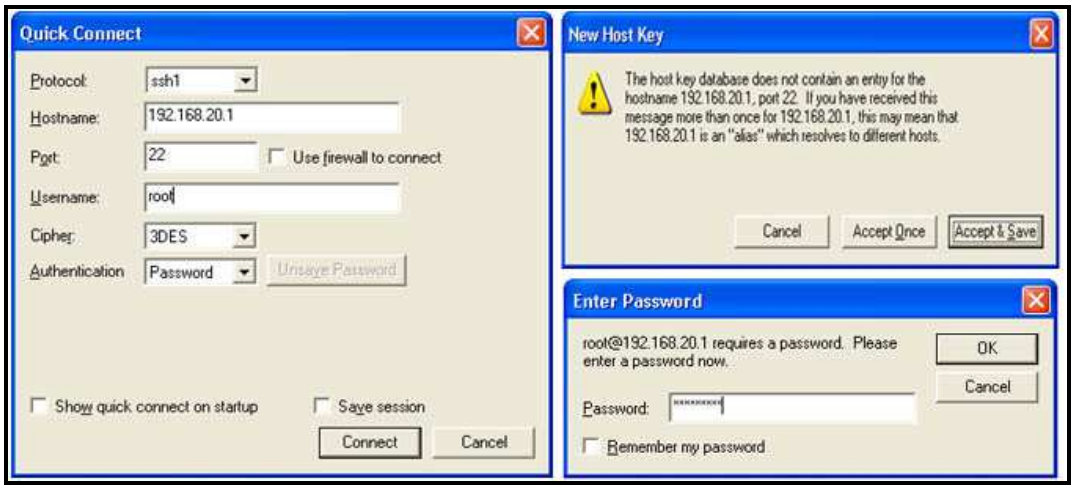
   ```
   sshd (pid 2882 2826 1136) is stopped
   ```

   maka *service* SSH *daemon* pada *router* belum dijalankan pada saat *starting up*, baris perintah untuk menjalankan *service* SSH adalah:

   ```
   [root@rh-router root]# service sshd start
   Starting sshd:[  OK  ]
   ```

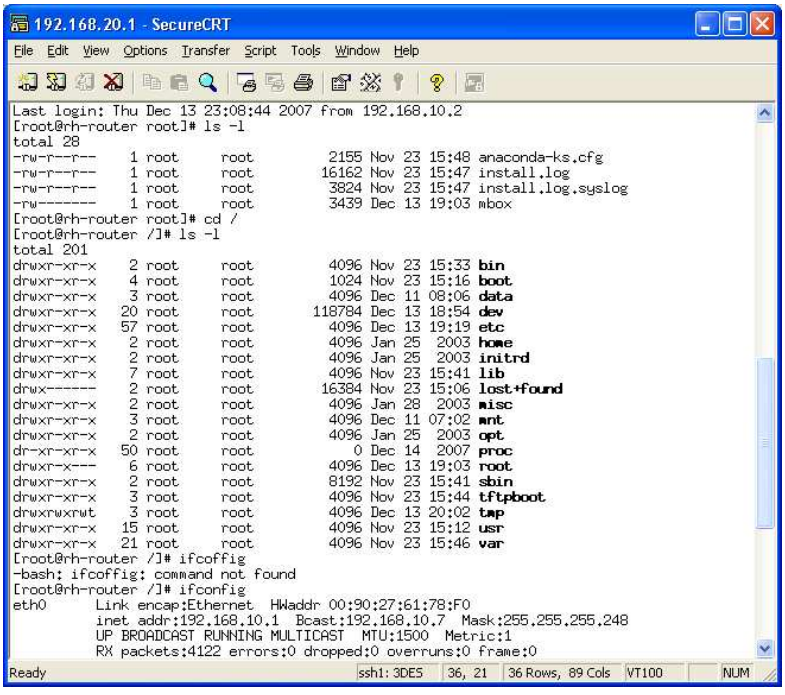
2. Pastikan bahwa *host* yang akan di-*remote* diketahui IP *address*-nya.

   Jika *SSH daemon server* telah aktif, maka tahap selanjutnya adalah dengan memulai koneksi melalui *SSH client*. Namun untuk me-*remote router* yang harus diketahui adalah alamat salah satu *interface* dari *router* tersebut. Pada *SecureCRT* ambil *Quick Connect*, lalu isikan alamat *ip address* router yang ada pada kolom *hostname*. Isikan *username* dengan *"root"* lalu click *connect*. *SSH server* kemudian akan men-*generate key* untuk *authenticate host*. Setelah *key* yang telah di-*generate* diterima, maka koneksi *host-to-host* telah berhasil dilakukan, masukan *password user root* dan *router* dapat dikonfigurasi melalui *console SecureCRT* pada *PC client*. Gambar 3.3 menunjukan koneksi SSH pada PC *router* menggunakan *SecureCRT*.



Gambar 3.3. Koneksi SSH menggunakan SecureCRT 4.0

3. Setelah koneksi berjalan, *router* dapat dikonfigurasi melalui *console*.



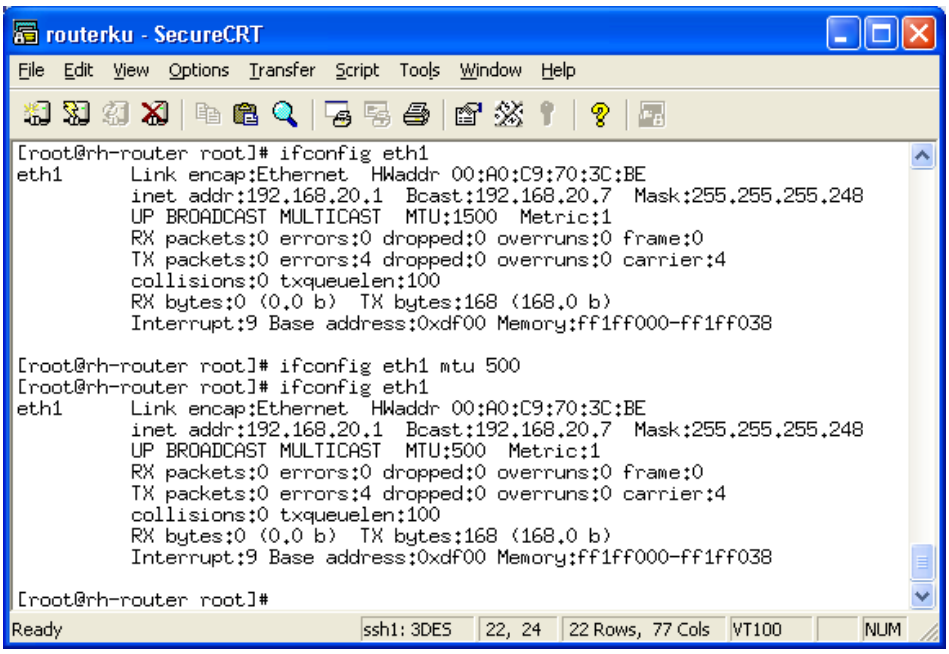
Gambar 3.4. Perintah Unix pada console SecureCRT 4.0

3.3.6 Sistem konfigurasi MTU

Dalam melakukan perubahan konfigurasi MTU diperlukan konfigurasi pada setiap *path-path* yang dilalui oleh paket. Perubahan konfigurasi diperlukan pada tiap *interface* yang ada *point-to-point*, baik yang ada pada *router* maupun pada *client*. Konfigurasi meliputi perubah setting pada sistem operasi yang dipakai pada *router* maupun *client*.

3.3.6.1 Sistem konfigurasi MTU pada PC router

Untuk mengubah konfigursi MTU yang ada pada PC router dapat dilakukan dengan me-*remote router* menggunakan *console* SSH *client Secure CRT* 4.0. Perubahan dilakukan pada tiap *interface* yang ada pada PC router. Perubahan ini dapat dilakukan dengan perintah "*ifconfig [interface name] mtu [mtu packet size]*", seperti yang terlihat pada gambar 3.6:

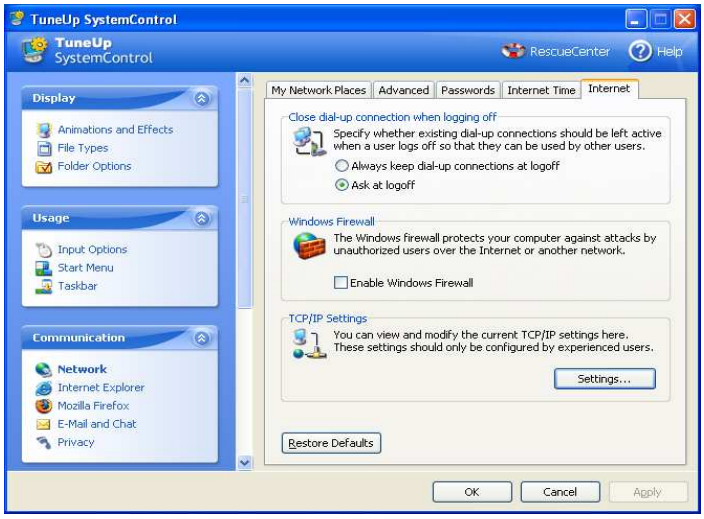


Gambar 3.6 Konfigurasi MTU pada PC router

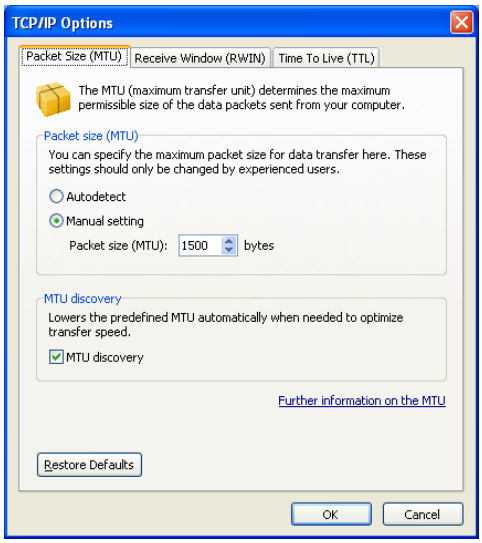
3.3.6.2 Sistem konfigurasi MTU pada PC client

Untuk mengubah *setting* MTU pada *Windows* digunakan *TuneUp Utilities 2006*. Lihat pada pilihan *TuneUp System Control*, lalu pilih *networks*. Dalam tab *Internet*, ambil pilihan TCP/IP setting dan set *Maximum Frame Sizes*. konfigurasi

MTU *default* adalah 1500 *bytes*. Nilai MTU yang dapat diubah antara 40 *bytes* sampai dengan 1500 *bytes*.



Gambar 3.7 Gambar *TuneUp Utilities 2006 (System Control)*



Gambar 3.8 Seting *Maximum Frame Size* untuk mengubah nilai MTU

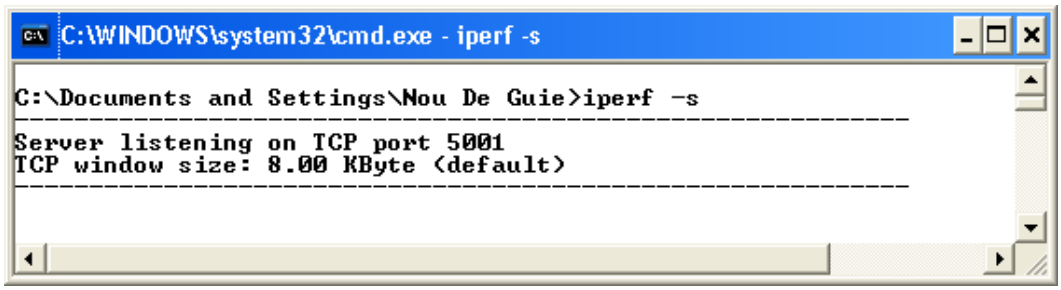
3.3.7 Penggunaan Aplikasi Simulasi

Beberapa aplikasi simulasi sebenarnya sama dengan aplikasi lainnya. Aplikasi tersebut biasanya di-*install* atau di-*copy* saja ke dalam *hardisk*, dan beberapa aplikasi biasanya dibutuhkan untuk berjalan pada *dua end-point* atau *client*. Berikut keterangan untuk tiap aplikasi yang membutuhkan cara penggunaan.

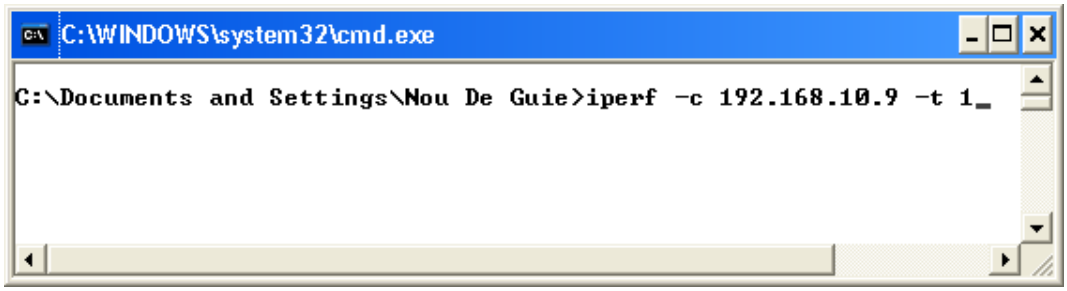
3.3.7.1 Aplikasi Iperf Versi 1.7.0

Aplikasi *Iperf* merupaka alat pengukur performa *network* yang dijalankan menggunakan perintah DOS *command line*. *Iperf* diaktifkan dengan masuk ke dalam direktori dimana file *iperf.exe* berada. Perintah-perintah yang digunakan dalam penggunaan *Iperf* dapat dilihat pada manual *Iperf*. *Iperf* harus dijalankan pada tiap PC *client*. *Iperf* pertama bertindak sebagai *server*, sedangkan *iperf* lainnya bertindak sebagai *client*. Setelah *iperf server* diaktifkan, maka pengiriman paket cukup dilakukan disisi *iperf* yang bertindak sebagai *client*.

Pada gambar 3.9 dapat dilihat *iperf* yang dikonfigurasi sebagai *server* dan pada gambar 3.10 adalah gambar *iperf* yang dikonfigurasi sebagai *client*.



Gambar 3.9 *Iperf* yang bertindak sebagai *server*



Gambar 3.10 *Iperf* yang bertindak sebagai *client*

3.3.7.2 Aplikasi Qcheck

Aplikasi *Qcheck* berfungsi untuk mengukur *Response Time*, *Throughput*, *Streaming* dan *Traceroute* dengan men-*generate* data dan mengirimkannya ke *end point* yang dituju. Pada simulasi ini aplikasi *Qcheck* digunakan untuk mengukur *throughput* transfer *file* berbasiskan protokol UDP dan *throughput streaming* dari *client* ke *server* melalui PC *router*. Fungsi *streaming* pada *Qcheck* terdiri dari

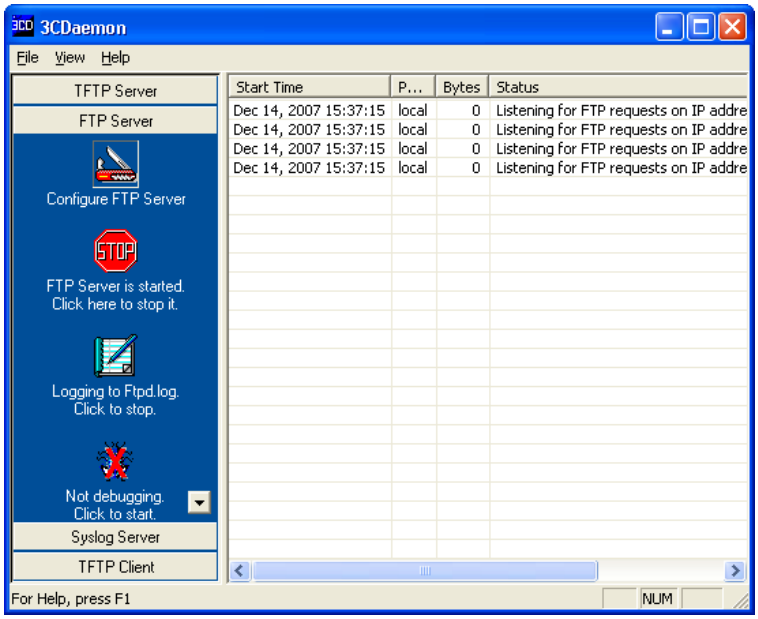
pengaturan besar *data rate* dalam satuan *Kbps* atau *Mbps* dan waktu durasi *streaming* bekisar dari 5 detik sampai 30 detik

Gambar 3.11 Qcheck dari Ixia

Pada pengukuran *throughput streaming* opsi yang perlu diisikan yaitu *data rate* sebesar 1000 Kbps dan *duration* selama 5 detik.

3.3.7.3 FTP Server

FTP (*File Transfer Protocol*) merupakan salah satu fasilitas yang berguna untuk mentransfer data dari FTP *server* ke FTP *client*. transfer data yang dimaksud dapat berupa *download* maupun *upload file* dari dan ke FTP *server*. Sebagian FTP *server* mensyaratkan untuk memiliki *login* dan *password* untuk dapat mengakses ke dalamnya, tetapi ada juga yang tidak mensyaratkan demikian. FTP *server* ini disebut *Server FTP Public* atau *Anonymous FTP*, dan dapat menggunakan *anonymous* untuk login dan *alamat email* untuk passwordnya.



Gambar 3.12 FTP Server

Untuk penggunaan FTP server, tidak diperlukan konfigurasi yang terlalu banyak. Cukup dengan meng-*install* lalu menjalankannya saja FTP *server* telah dapat berjalan dengan baik. Namun ada satu hal yang harus diperhatikan, yaitu *setting user directory*. *User directory* adalah tempat dimana nantinya FTP *server* akan meletakan direktori yang aktif pada saat FTP *client* terhubung. Maka *user directory* harus di-*setting* pada direktori yang sesuai, agar pada saat FTP *client* melakukan koneksi, *user* yang *login* dapat mengakses direktori yang dituju.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Optimasi PC Router

Bagian ini menjelaskan implementasi dalam mengoptimalkan kinerja dari *operating system* pada *PC router*. *Operating system* yang akan dioptimasi adalah *Linux Redhat 9.0*. Optimasi yang dilakukan meliputi optimasi pada *kernel* dan *service-service* yang berjalan serta optimasi pada *device* dan perangkat jaringan lainnya. Kemudian akan dilakukan pengujian terhadap *PC router* sebelum dan sesudah dioptimasi. Hasil pengujian nantinya akan dibandingakan, sehingga terlihat apakah ada peningkatan performa setelah *PC router* dioptimasi. Pengujian meliputi pada dua protokol, yaitu TCP (*connection oriented*) dan UDP (*connectionless*). Kemudian pada bagian hasil akan menyajikan grafik dari statistik pengukuran simulasi beserta analisa hasil sehingga dapat digunakan untuk memudahkan pengambilan kesimpulan. Hasil pencatatan dan pengukuran dalam bentuk tabel-tabel selengkapnya terdapat pada bagian lampiran.

4.2 Batasan Optimasi Dan Simulasi

Optimasi yang dilakukan pada PC router adalah mengoptimasi seluruh perangkat pengendali pada sistem operasi. Sistem operasi yang dioptimasi adalah sistem operasi Linux yang merupakan turunan dari sistem operasi Unix, sedangkan distribusi yang digunakan adalah Red Hat 9.0. Optimasi secara keseluruhan dilakukan pada sistem operasi tersebut, yaitu optimasi pada kernel, service, dan MTU (Maximum Transfer Unit). Pemilihan kernel dan service untuk optimasi dilakukan karena secara karakteristik, sistem operasi yang digunakan mendukung hal itu. Optimasi pada nilai MTU juga diperlukan, yaitu untuk melihat kinerja dari transfer data jika dilakukan perubahan MTU.

Kemudian simulasi yang dilakukan merupakan simulasi dari satu end-point menuju ke end-point yang lain. End-point dalam hal ini merupakan dua PC

workstation yang mengirimkan data melewati PC router. Karena keterbatasan perangkat simulasi pada LAN tidak dapat dilakukan pada kondisi yang sebenarnya dan pada traffic data yang sibuk. Namun diharapkan dengan dukungan aplikasi-aplikasi simulasi diharapkan sudah dapat menunjukan korelasi antara parameter-parameter yang diteliti. Keterbatasan pada aplikasi simulasi juga membuat penulis hanya mengukur througput pada jaringan menggunakan protokol aplikasi FTP dan streaming pada protokol TCP dan UDP.

4.3 Optimasi PC Router

Sebelum pengujian dilakukan, terlebih dahulu dilakukan pengujian pada PC router yang belum dioptimasi yang kemudian didokumentasikan terlebih dahulu. Sebelum data pembanding dibandingkan dengan data pertama, terlebih dahulu PC router dioptimasi. Ada beberapa hal yang dioptimasi pada Operating System PC router. Point-point tersebut akan dijelaskan dibawah ini.

4.3.1 Optimasi Kernel

Kernel merupakan bagian terpenting dari sistem operasi *Linux*. *Kernel Linux* terdiri dari beberapa bagian penting, seperti manajemen proses, manajemen memori, *hardware device drivers*, *filesystem drivers*, manajemen jaringan dan lain-lain. *Kernel* juga menyediakan sekumpulan layanan yang digunakan untuk mengakses *kernel* yang disebut *system call*. *System call* ini digunakan untuk mengimplementasikan berbagai layanan yang dibutuhkan oleh sistem operasi. Namun tidak semua dari layanan pendukung yang menyusun sebuah kernel tersebut diperlukan, misalnya kernel memiliki sekumpulan *modules* dan *device driver* dari banyak *vendor* didalamnya, namun hanya beberapa *modules* dan *device driver* saja yang akan dipakai sebagai komponen pendukung. Kita dapat mengoptimalkan kinerja dari system agar kernel berjalan dengan efisien, yaitu dengan meng-*compile kernel*. *Compile kernel* dilakukan untuk membuat ukuran *kernel* menjadi se-*optimal* mungkin. Dengan ukuran yang optimal, *kernel* dapat dengan cepat memekarkan system dan langsung me-*load* nya kedalam *memory*

untuk diproses. Sehingga sistem dapat berjalan dengan efisien dan hal ini tentunya tidak membebani sistem.

Compile kernel adalah salah cara yang kita lakukan dalam mengoptimasi sistem operasi. Compile kernel adalah salah satu cara untuk mendapatkan system operasi yang ideal untuk system kita. Berdasarkan hasil *googling* penulis, *compile kernel* merupakan salah satu cara bagi para pengguna linux untuk mendapatkan hasil yang optimal untuk sebuah sistem yang solid sesuai dengan keinginan user tanpa diganggu oleh modul yang tidak diinginkan. Source kernel yang masih mentah dapat kita compile menjadi kernel yang baru dan lebih efisien.

4.3.1.1 Compile Kernel

Optimasi pada kernel yang akan lakukan adalah dengan mengcompile kernel. Compile kernel dapat dilakukan hampir di semua versi kernel linux, dari versi 2.2, 2.4 ,dan 2.6. Namun apakah kompilasi ulang kernel diperlukan? Berikut beberapa alasan mengapa dibutuhkan compile kernel.

1. Kernel baru biasanya dikeluarkan secara berkala, dimana antara satu kernel dengan kernel yang lain berbeda walaupun sangat minim. Meng-update kernel baru secara berkala sangat tidak dianjurkan, sampai versi kernel baru yang dikeluarkan benar-benar mengeluarkan *patch* yang penting dengan *security* dan *stability*.
2. Dahulu kernel tidak se-modular sekarang. Ini menggambarkan bahwa kernel lama memuat banyak module yang tidak diperlukan kedalam memory yang menambah beban kedalam sistem dan menambah bugs bagi module-modul yang tidak diperlukan. Recompile kernel untuk menghapus modul-modul yang tidak dibutuhkan memiliki keuntungan. Kernel baru, biasanya memuat modul kedalam memory bila dibutuhkan. Menghapus secara manual modul ini merupakan hal positif, bagaimanapun modul-modul yang tidak diperlukan ini tidak lagi dimuat oleh kernel.
3. *Recompile kernel* dibutuhkan apabila hardware baru ditambahkan kedalam system, sedangkanyang kernel lama tidak men-support.

Sebagai contoh, system dengan motherboard dual processor, tapi yang terpasang hanya satu processor saja. Jika dilain hari ada penambahan processor, kernel harus di-recompile ulang agar support dengan Symmetic Multi-Processing (SMP) sehingga processor kedua yang ditanamkan berjalan dengan baik.

Sebelum mengkompilasi kernel, ada baiknya kita menentukan terlebih dahulu, kira-kira format kernel yang bagaimana yang sesuai dengan kebutuhan. Sebagai contoh jika PC sering mengganti harware dengan hardware lain yang berlainan vendor maupun tipe, sistem kernel yang modular akan lebih cocok daripada sistem kernel yang monolitik (built-in)

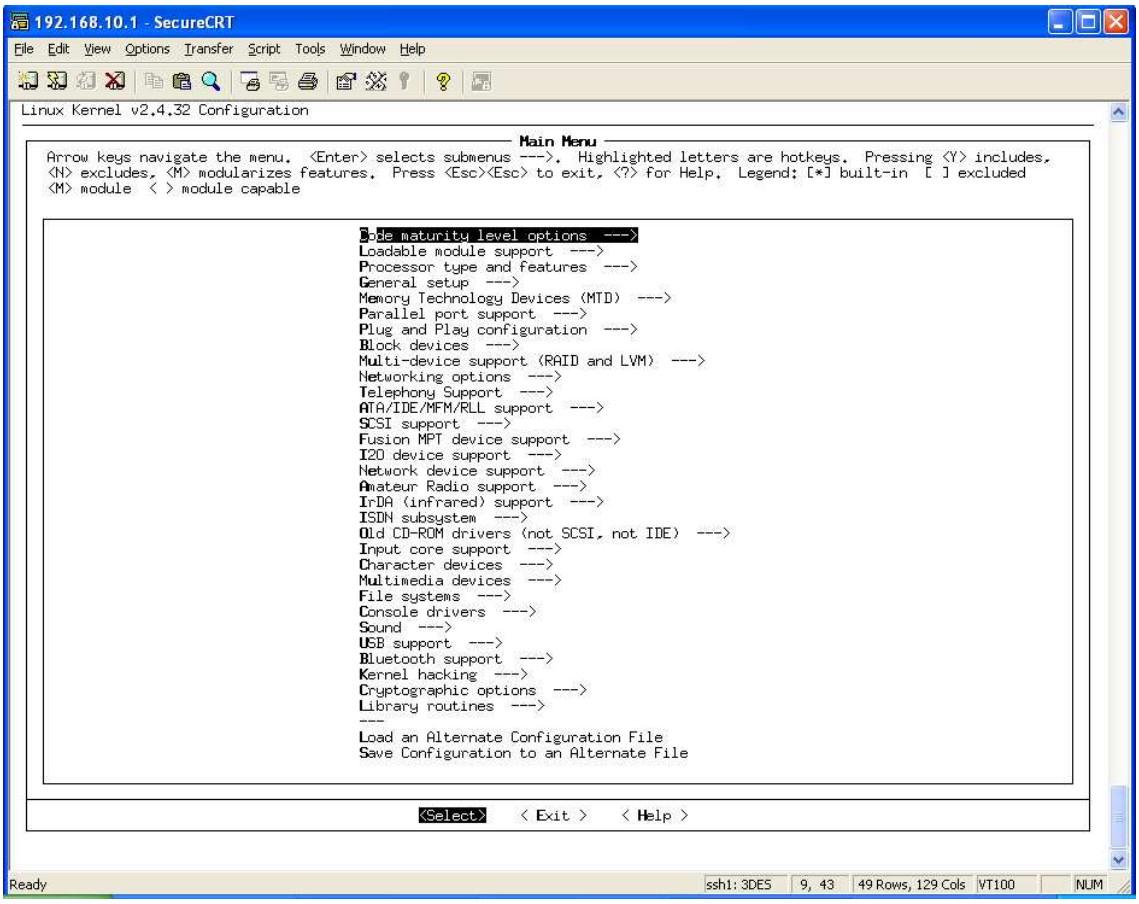
Sebelum kita meng-compile kernel ada beberapa tahapan yang harus kita lakukan, yaitu;

1. Pastikan versi kernel yang ada pada sistem operasi Linux kita, versi bawaan dari Linux Red Hat 9.0 yang digunakan dalam penelitian ini adalah kernel versi 2.4.20-8. Kita dapat melihat versi kernel dengan mengetikan perintah *uname -a* dari *console*.
2. Registrasi semua devices yang dibutuhkan oleh sistem menggunakan perintah *cat /proc/cpuinfo* untuk melihat processor yang digunakan, lspci dan dmesg untuk melihat devices yang ada, misalnya ethernet card, chipset, vga card, serta soundcard. Dan perintah *mount* untuk melihat file system yang digunakan pada PC router.
3. Download kernel versi 2.4.32 dari *www.kernel.org* , lalu extract ke direktori /usr/src/linux-2.4.32 (*tar -jxvf linux-2.4.32.tar.bz2*)
4. Masuk kedirektori /usr/src/linux-2.4.32
5. Untuk pemilihan konfigurasi kernel yang akan di-*compile* gunakan perintah *make menuconfig*.

4.3.1.2 Setting Parameter Kernel

Kernel dapat dikonfigurasi sesuai dengan keinginan kita. Konfigurasi yang ingin kita dapatkan dari setting parameter kernel adalah optimum untuk router atau network. Konfigurasi ini melibatkan pengurangan beberapa element dan

modul yang tidak dibutuhkan untuk PC router. Berikut hasil setting parameter pada kernel 2.4.32.



Gambar 4.1 Main Menu dari Konfigurasi Linux Kernel

Dari gambar diatas, ada beberapa option yang akan kita disable, yaitu;

1. Code maturity level options

   Merupakan kumpulan Network drivers, file systems, network protocols, dan modul lainnya yang masih dalam tahap development atau percobaan.

2. Loadable module support

   Karena akan membuat kernel yang monolitik yang tidak mengandung modul, maka option ini tidak akan dimasukan kedalam kernel.

3. Memory Technology Devices (MTD)

   MTD merupakan flash, RAM and chips yang biasa mendukung file system pada embedded devices.

4. Parallel port support

   Karena router tidak memerlukan port pararel untuk printer, maka option ini akan kita disable.

5. Multi-device support (RAID and LVM)

   Berfungsi untuk RAID dan LVM (Logical Volume Manager).

6. Telephony Support

   Berfungsi untuk memberikan modules voip cards jika devices tersedia.

7. SCSI support

   Berfungsi jika ingin menggunakan SCSI device seperti SCSI hardrive, SCSI tape, dan SCSI cdrom.

8. Fusion MPT device support

   Tidak ada option pilihan.

9. I2O device support

   The Intelligent Input/Output option.

10. Amateur Radio support

    Berfungsi sebagai radio amatir pada Linux.

11. IrDA (infrared) support

    Berfungsi sebagai infrared support.

12. ISDN subsystem

    Berfungsi sebagai Integrated Services Digital Networks. Sering digunakan untuk mengkoneksikan diri ke ISP dengan SLIP atau PPP).

13. Old CD-ROM drivers (not SCSI, not IDE)

    Berfungsi jika anda memiliki CD ROM lama yang bukan SCSI ata IDE.

14. Input core support

    Berfungsi untuk USB Human Interface Device (HID) support.

15. Multimedia devices

    Berfungsi sebagai support untuk audio/video capture dan overlay devices serta FM radio card.

16. Sound

    Berisi device dan modules driver bagi soundcard.

17. USB support

    Modul pendukung USB (Universal Serial Bus).

18. Bluetooth support

    Modul pendukung untuk blutooth dan pairing devices.

19. Kernel hacking

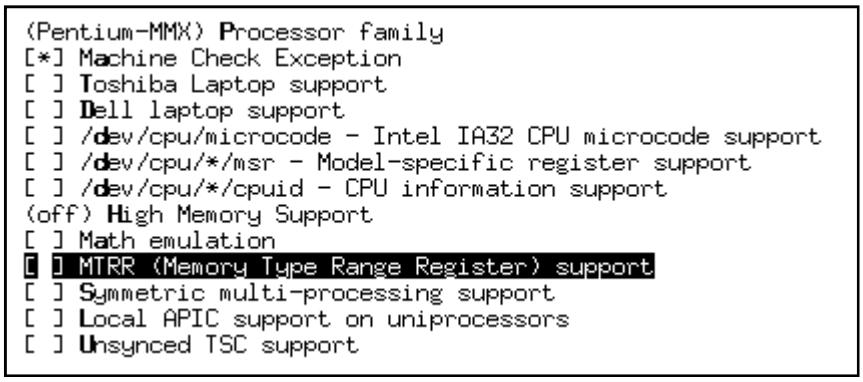
    Modul pendukung untuk hack kernel dalam modus CLI.

20. Cryptographic options

    Modul pendukung untuk enkripsi Cryptographic.

Pemilihan option yang benar-benar kita butuhkan untuk proses routing dan transfer data akan membuat performa router akan menjadi semakin baik. Dalam hal ini pemilihan modules dan devices drivers yang kita butuhkan adalah;

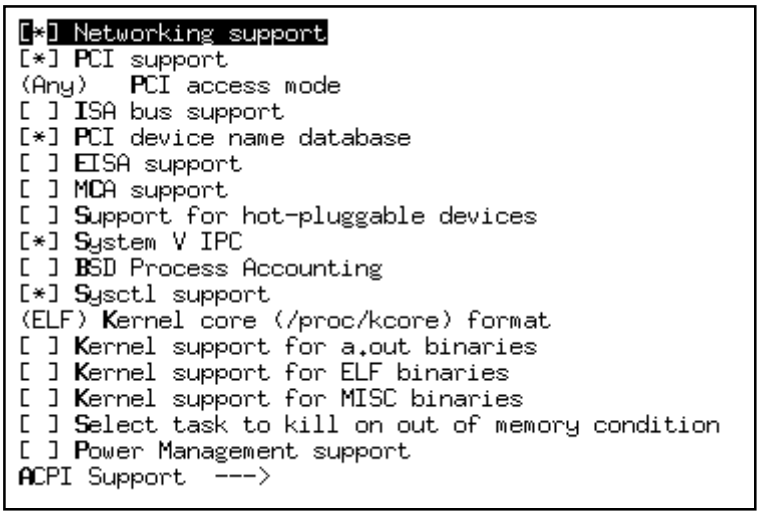
1. Processor type and features



Gambar 4.2 Konfigurasi pada Processor type and features

Identifikasi yang kita lakukan pada saat awal sebelum mengkonfigurasi kernel, pada processors didapatkan data bahwa processor adalah Pentium-MMX. Maka pada pemilihan processor family dapat kita pilih Pentium-MMX. Untuk High Memory Support kita pilih off, karena pemilihan tersebut akan membuat kernel membaca RAM yang lebih dari 1GB keatas. Math emulation juga jangan dipilih, karena ini merupakan emulasi digital pada 386 & 486 SX pada coprocessor, yang akan menyebabkan gangguan pada processor diatasnya. Dan SMP perlu didisable, karena kita tidak menggunakan multi-procesor disini.

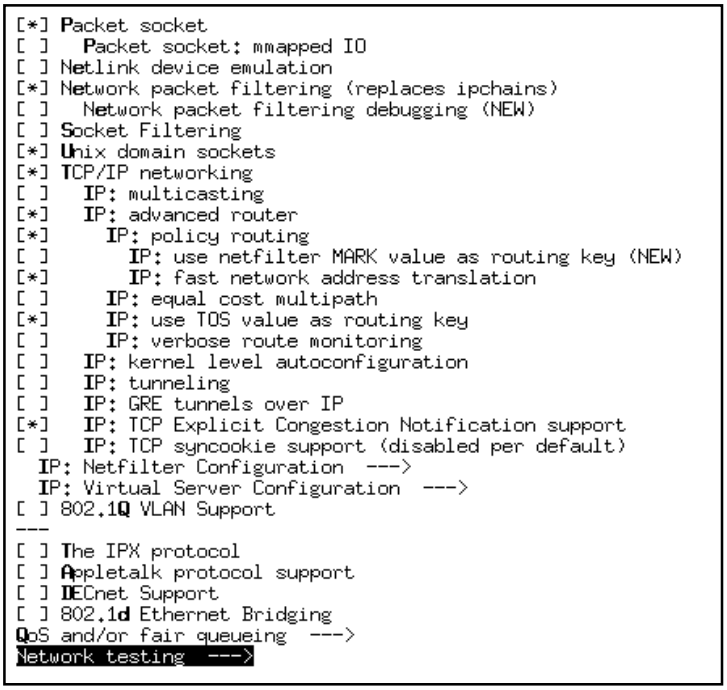
2. General setup



Gambar 4.3 Konfigurasi General Setup

Konfigurasi disini merupakan konfigurasi umum untuk *kernel*. Kita membutuhkan *Network Support, PCI Support,* serta *Sysctl Support*. Khusus untuk menjadikan *Linux Box* menjadi sebuah *PC router*, maka *Sysctl* harus diaktifkan, jika tidak PC router tidak akan mau mem-*foward* paket yang akan kita kirimkan.

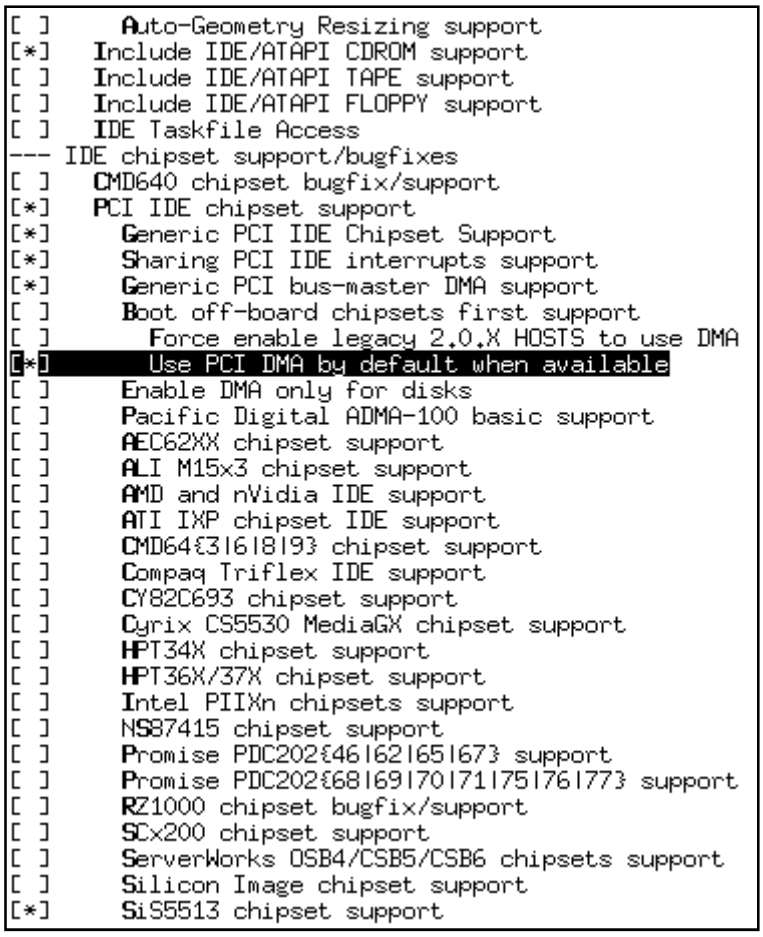
3. Networking options



Gambar 4.4 Konfigurasi Network Option

Pada networking option ada beberapa hal yang harus diperhatikan. Pada TCP/IP, advance router harus di-*enable* kan sebagai sebuah *router*. Menggunakan TOS (*Type Of Service*) sebagai routing key juga akan membuat dampak yang pada router. Untuk securitas router anda jangan lupa untuk checklist *network packet filetering* pada *kernel konfiguration interface.*

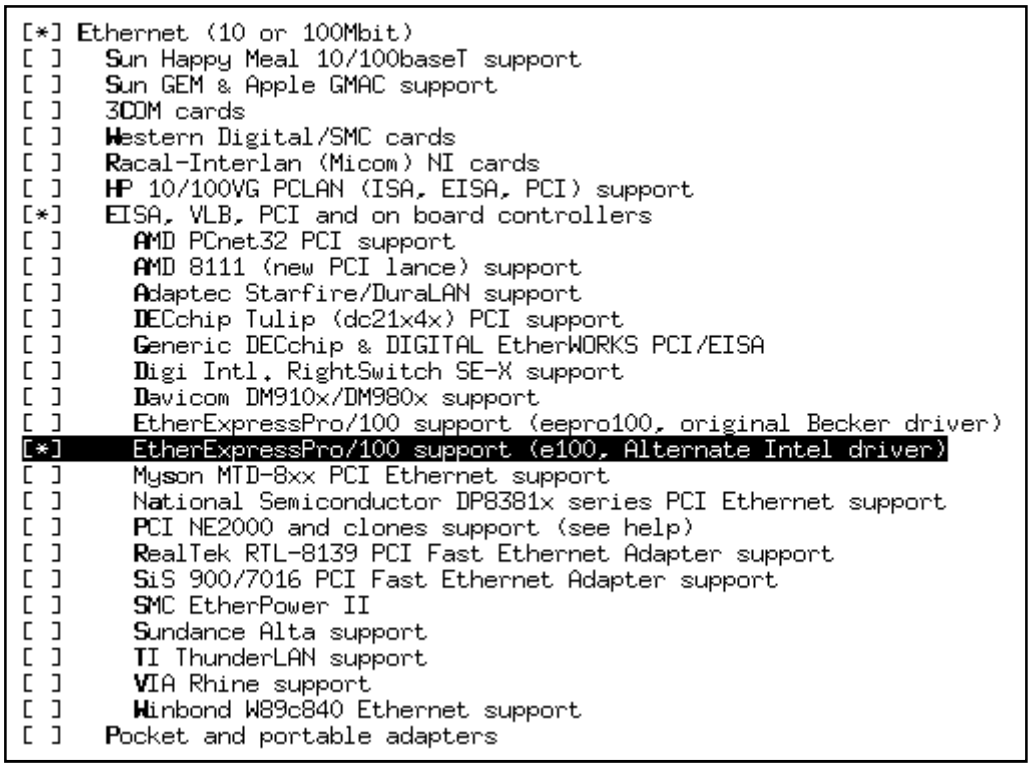
4. ATA/IDE/MFM/RLL support



Gambar 4.5 Konfigurasi ATA/IDE devices

Disini terdapat pemilihan pada devices / chipset support untuk IDE. Pada proses identifikasi awal menggunakan *"lspci" command* didapat *SIS5513 chipset* yang digunakan pada sistem. Modul ini digunakan oleh chipset untuk berkomunikasi dengan hardrive dan device lain yang menggunakan antar muka IDE.

5. Network Devices support



Gambar 4.5 Konfigurasi Network Devices support

Pada option ini kita diperkenankan untuk memilih devices drivers yang akan disupport oleh system. Pada identifikasi awal menggunkaa "*dmesg | grep eth" command*, didapat kan data bahwa network card yang dipakai adalah Intel(R) PRO/100 Network Connection dengan modul e100. Pastikan untuk memasukan modul e100, dan disable sisa dari network device drivers yang ada.

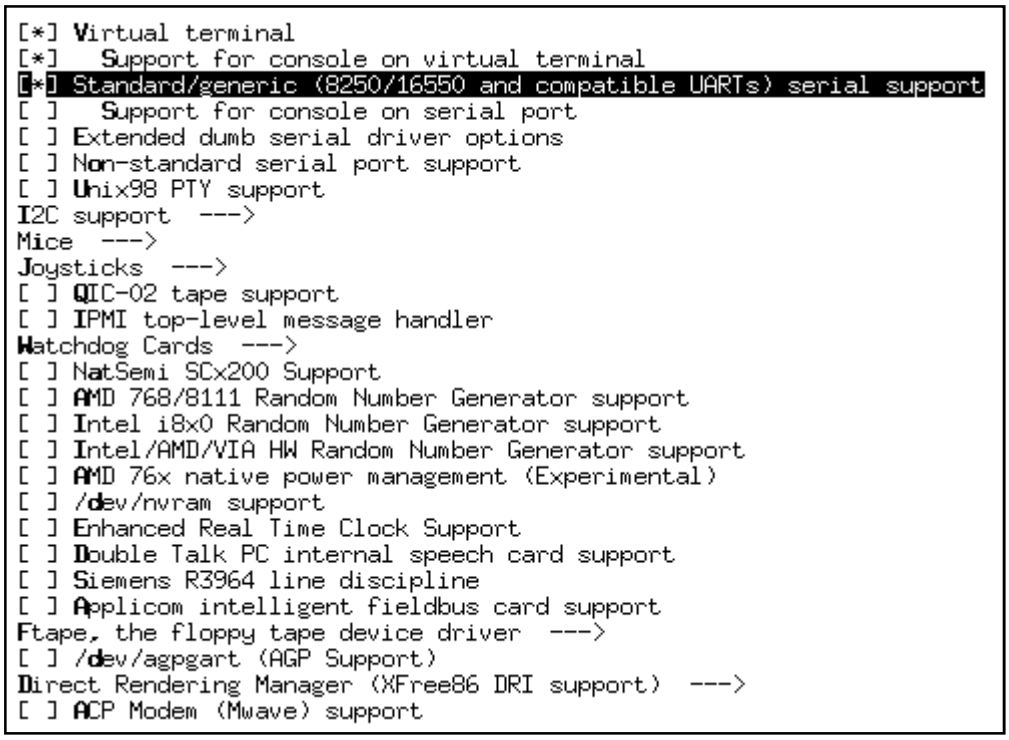
6. File systems

```
[ ] Quota support
[ ] Kernel automounter support
[ ] Kernel automounter version 4 support (also supports v3)
[ ] Reiserfs support
[*] Ext3 journalling file system support
[*]   JBD (ext3) debugging support
[ ] DOS FAT fs support
[ ] Compressed ROM file system support
[*] Virtual memory file system support (former shm fs)
[*] ISO 9660 CDROM file system support
[*]   Microsoft Joliet CDROM extensions
[ ]   Transparent decompression extension
[*] JFS filesystem support
[ ]   JFS debugging
[ ]   JFS statistics
[ ] Minix fs support
[ ] FreeVxFS file system support (VERITAS VxFS(TM) compatible)
[ ] NTFS file system support (read only)
[ ] OS/2 HPFS file system support
[*] /proc file system support
[ ] QNX4 file system support (read only)
[ ] ROM file system support
[*] Second extended fs support
[ ] System V/Xenix/V7/Coherent file system support
[ ] UDF file system support (read only)
[ ] UFS file system support (read only)
[ ] XFS filesystem support
Network File Systems  --->
Partition Types  --->
Native Language Support  --->
```

Gambar 4.7 Konfigurasi File System

Linux banyak sekali mendukung file system, diantaranya Ext2, Ext3, JFS,XFS, UDF,bahkan Fat dan NTFS. Namun dalam sebuah router, hanya beberapa yang diperlukan yaitu Ex2, Ext3 dan JFS (IBM's Journalling Filesystem). Khusus untuk mendukung routing, maka /proc file system support harus di-enable kan. Jika tidak maka routing tidak akan terjadi.

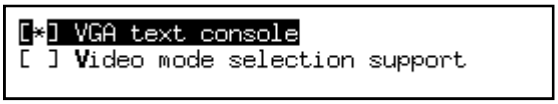
7. Character devices



Gambar 4.8 Konfigurasi Character Devices

Option ini digunakan untuk Virtual Terminal yang akan meremote console linux menggunakan SSH ataupun Telnet. Checklist pada Virtual Terminal jika sistem ingin diremote dari PC lain melalui network.

8. Console drivers



Gambar 4.9 Konfigurasi Console drivers

Untuk penggunaan console, check VGA text console untuk default text console pada modus VGA.

Setelah itu kita dapat meng-compile kernel dengan perintah *make bzImage*. Namun penulis menemukan scrip yang dapat membantu untuk menghitung waktu yang dibutuhkan pada saat compile kernel berlangsung, yaitu ;

echo "Mulai: $(date)" > waktu; make bzImage; echo "Selesai: $(date)" >> waktu

setelah kernel di-compile maka kita dapat mengetahui waktu yang dibutuhkan untuk compile kernel dengan mengetikan perintah "*cat waktu*" dan hasilnya:

[root@rh-router linux]# cat waktu
Mulai: Tue Jan 15 17:42:52 WIT 2008
Selesai: Tue Jan 15 18:28:04 WIT 2008

4.3.2 Service

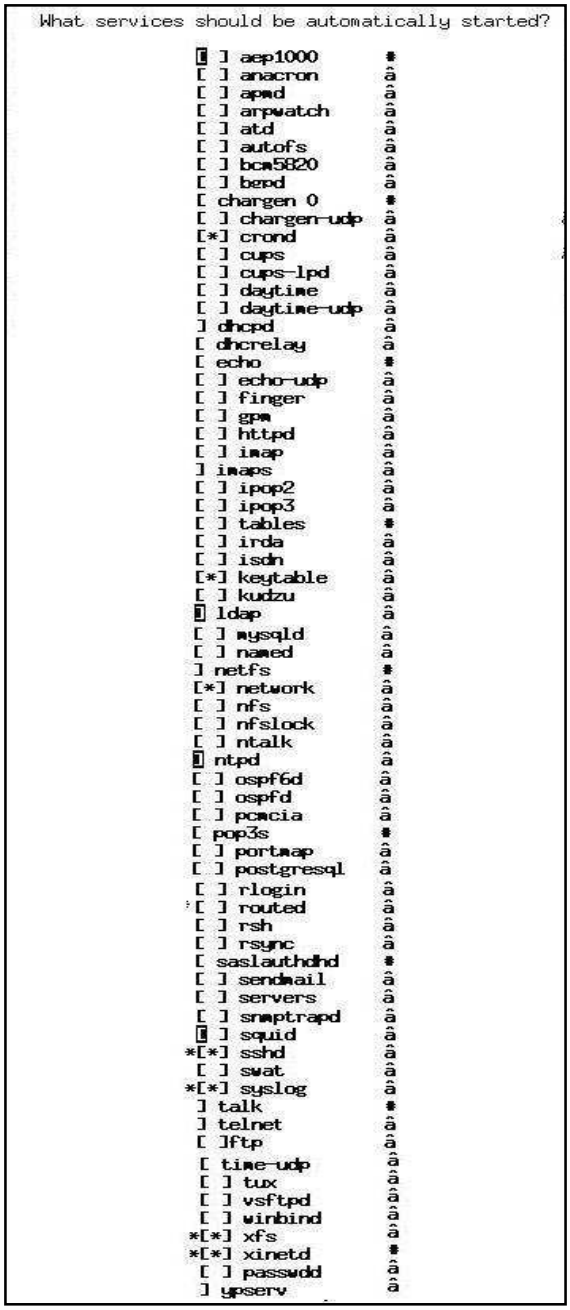
Pada saat default installasi dari *Linux Red Hat 9.0* (*Shrike Code*), ada beberapa service yang dijalankan secara otomatis pada saat sistem dihidupkan. Beberapa service tersebut berguna, dan beberapa lainnya tidak. Berdasarkan fungsi *Linux Box* yang kita miliki digunakan hanya sebagai router, maka banyak service yang benar-benar tidak diperlukan dan harus di-*disable* pada saat start up. Tujuan utama men-*disable service* yang tidak diperlukan adalah *management memory*. Memory di-*design* sebagai *temporary buffer*, dimana dengan space memory yang lebih besar maka proses pengiriman data pun dapat berlangsung dengan cepat. Process untuk mengaktifkan maupun untuk mematikan suatu service dapat dilakukan melalui command :

1. service
   service < option > | --status-all | [service_name [command | --full-restart]]
2. chkconfig
   chkconfig [--level <levels>] <name> <on|off|reset>)
3. ataupun melalui *ntsysv* yang berbentuk table yang dengan mudah kita enable atau disable.

Beberapa service yang aktif dan tidak kita butuhkan, yaitu

1. *anacron* : menjalankan pekerjaan cron jika terlewati oleh cron.
2. *atd* : menjalankan perintah spesifik yang telah dibuat pada waktu yang telah diset oleh pengguna.
3. *autofs* : auto mounter untuk Unix file system.
4. *cups* : menjalankan Common UNIX Printing System pada saat startup / shutdown.
5. *crond* : crond merupakan standard Unix program yang menjalan kan program pada waktu yang ditentukan. Sama halnya dengan task scheduler pada windows.

6. *gpm* : linux mouse support for console, berguna jika menjalankan program seperti *Midnight Commander*.
7. *isdn* : berfungsi untuk mengaktifkan ISDN *service*.
8. *kudzu* : mengkonfigurasi dan menjalankan probing pada hardware baru.
9. *netfs* : mount dan unmount NFS (*Network File System*), Samba (*Windows*), dan NCP (*Netware*).
10. *nfslock* : menyediakan fungsi locking pada Network File System (NFS).
11. *pcmcia* : menyediakan fungsi untuk PCMCIA *cards*.
12. *portmap* : me-*manage* RPC koneksi pada protokol NFS dan NIS.
13. *rhnsd* : *check update* dan notifikasi dari *Red Hat Servers*.
14. *raw devices* : biasa digunakan pada aplikasi *Oracle*.
15. *sendmail* : *Mail Transport Agent* yang biasa digunakan untuk mengirim dan menerima *e-mail*.
16. *sgi-fam* : file *monitoring*, akan memberikan notifikasi kepada user bila ada perubahan.

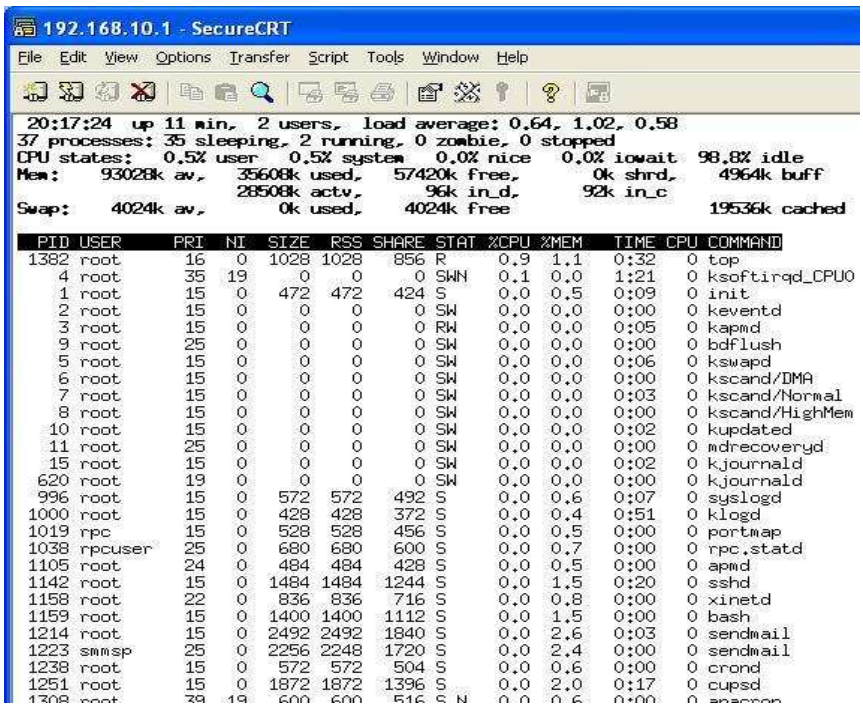


Gambar 4.10 Konfigurasi Service menggunakan ntsysv

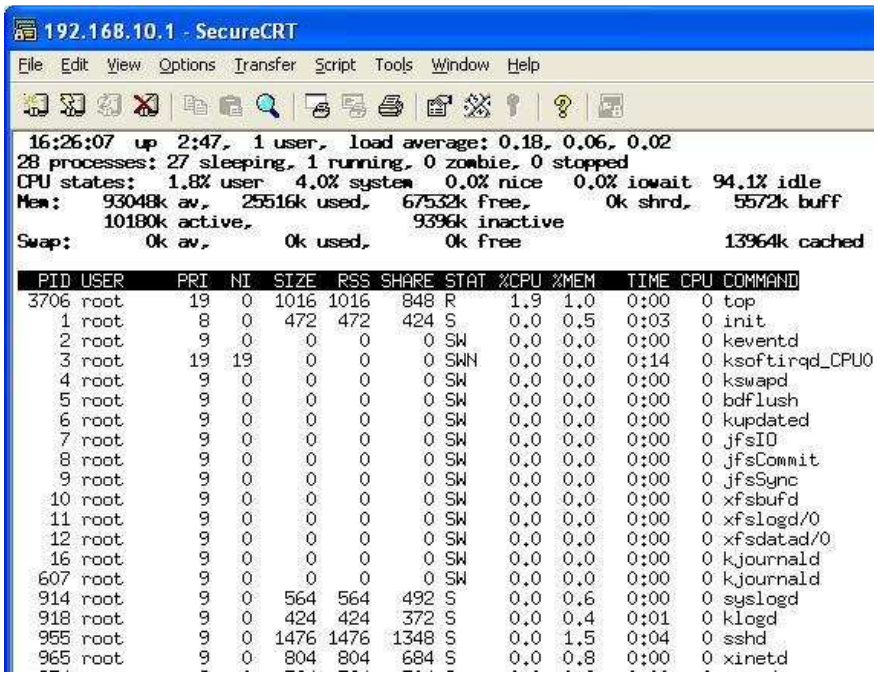
4.3.3 Perbandingan Resources

Setelah dioptimasi apakah terlihat peningkatan pada resource PC router? Hal itu dapat dibuktikan dengan membandingkan resource yang ada pada PC router. Untuk itu kita gunakan program top, seperti aplikasi task manager pada

windows. Hasilnya pada saat *idle*, PC router yang telah kita *compile* dan kita optimasi *service*-nya memang terdapat perbedaan. Seperti hasil *capture* gambar dibawah ini.

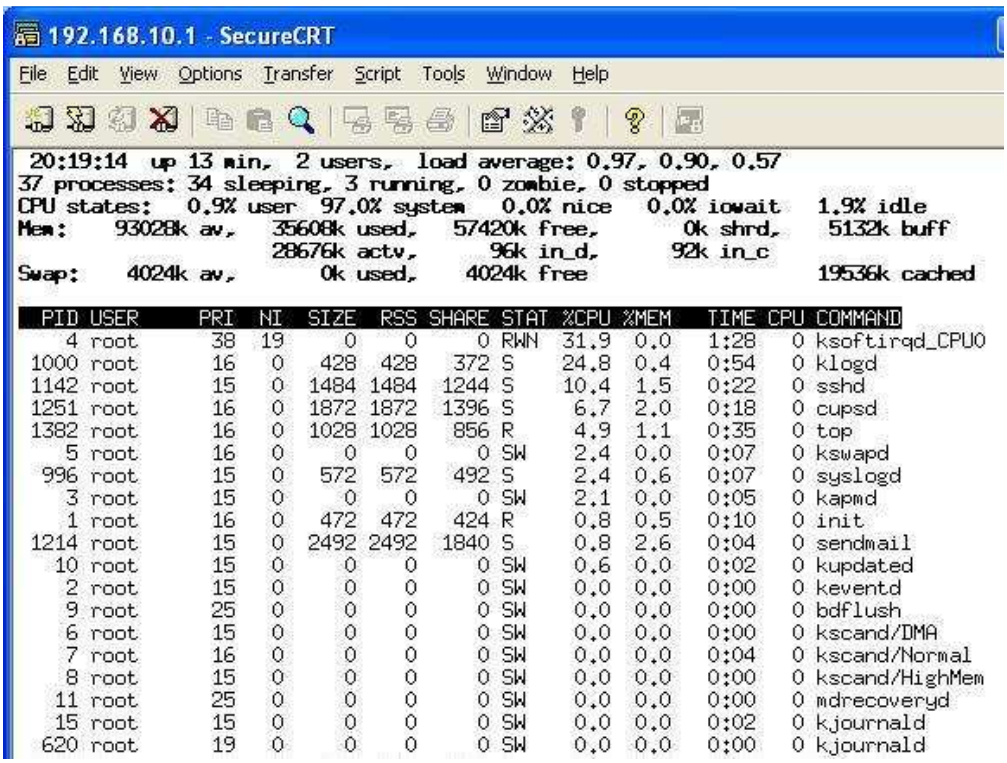


Gambar 4.11 Service Running with top before optimization

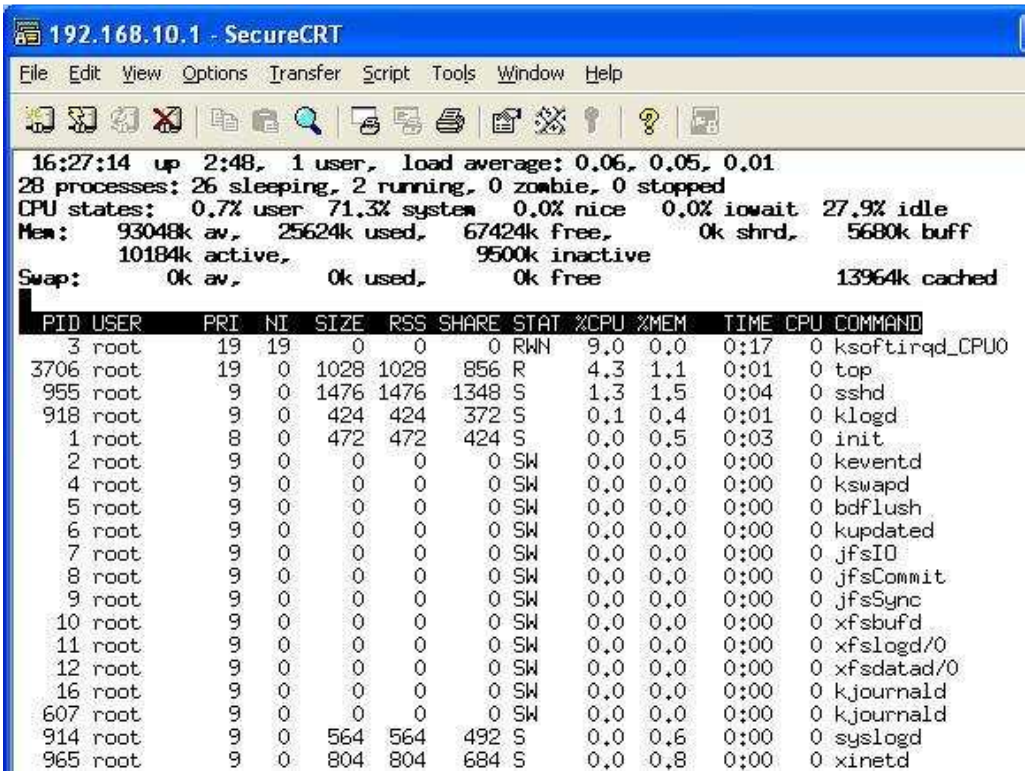


Gambar 4.12 Service Running with top after optimization

Pada saat idle terlihat bahwa ada peningkatan pada free memory. Pada saat sebelum dioptimasi nilai penggunaan memory berkisar pada angka 35608 KB, sedangkan pada saat setelah dioptimasi terjadi penurunan sekitar ±10MB pada 25516 KB. Pada saat sebelu optimasi didapat running process berjumlah 37 process, sedangkan pada saat setelah optimasi didapat penurunan running process berjumlah 28 process. Kemudiaan kita bandingkan lagi load CPU pada saat data ditransfer, hasil juga terdapat perbedaan. Seperti hasil capture pada gambar dibawah ini.



Gambar 4.13 top before optimization



Gambar 4.14 top after optimization

Terlihat perbandingan pada saat router di-load dengan data. Penggunaan CPU untuk transfer data pada linux pada saat sebelum optimasi hampir mencapai nilai maximum, yaitu pada kisaran 93% nilai penggunaan CPU. Sedangkan pada PC router yang telah dioptimasi penggunaan CPU lebih rendah, berkisar antara 71%. Perbandingan diantara keduanya mencapai 20% penggunaan CPU. Sebuah hasil yang baik, mengingat hanya dua host yang dipakai sebagai pengujian.

4.4 Pengujian Transfer Data pada PC Router

Pengujian pada PC router melibatkan protokol TCP dan UDP. TCP dengan pengiriman paket yang digaransi, sedangkan UDP dengan pengiriman paket yang tidak digaransi. Pengujian dilakukan menggunakan aplikasi FTP, Iperf, dan Qcheck. Pengujian juga melibatkan nilai MTU yang biasanya ada pada network devices. MTU (Maximum Transfer Unit) adalah parameter satuan transfer data maksimal dalam *byte* oleh suatu perangkat jaringan atau *physical network interfaces*. Tujuannya adalah untuk mengetahui apakah besar MTU mempengaruhi hasil transfer data yang kita lakukan. Besar MTU pada setiap

media berbeda-beda, pada teknologi *Ethernet* nilai default MTU yang digunakan adalah 1500 *bytes*. Simulasi MTU dilakukan pada besar MTU 150, 300, 500, 1000, 1500 *bytes* dengan durasi 10 detik dan dilakukan sebanyak 10 kali iterasi. Berikut ringkasan pengujian dan simulasi transfer data pada PC router yang dilakukan oleh penulis.

4.4.1 Simulasi pada protokol TCP dengan Iperf

Pada BAB III telah dijelaskan bahwa simulasi untuk mengukur performa network pada protokol TCP dilakukan dengan Iperf. Iperf berkerja atas prinsip client-server. Host pertama bertindak sebagai server dan host kedua bertindak sebagai client. Pengukuran pada Iperf dilakukan sebelum dilakukan optimasi. Pengujian dilakukan sebanyak sepuluh iterasi. Hasil yang didapat dalam sepuluh iterasi dirata-ratakan, dan kemudian didapat nilai total atau nilai keseluruhan sebelum dilakukan optimasi pada sistem operasi. Kemudian dilakukan optimasi pada PC router. Setelah dioptimasi dilakukan juga pengujian yang sama pada saat sebelum dilakukan optimasi, yaitu dilakukan sebanyak sepuluh iterasi yang kemudian dimbil nilai rata-rata pengujian. Pengujian juga melihat seberapa besar pengaruh optimasi nilai MTU (Maximum Transfer Unit) pada PC router sebelum dan sesudah dilakukannya optimasi. Pengujian MTU ini akan diberlakukan untuk semua pengujian menggunakan Iperf, FTP, Qcheck (*throughput* & *streaming*) pada protokol TCP dan UDP.

4.4.1.1 Langkah-langkah Simulasi TCP dengan Iperf & MTU

Sebelum dilakukan pengujian ada beberapa konfigurasi yang harus kita lakukan untuk menentukan nilai dari parameter-parameter yang diteliti, langkah-langkah yang dilakukan antara lain :

1. Mengkonfigurasi Iperf sebagai server pada host pertama.
2. Mengkonfigurasi Iperf sebagai client pada host kedua.
3. Aktifkan IP routing pada PC router

4. Mencatat transfer rate yang dihasilkan pada transfer data menggunakan Iperf selama 10 second, dengan minimal pengambilan data 10 iterasi pada saat sebelum dan sesudah dioptimasi.
5. Khusus untuk optimasi MTU, simulasi menggunakan Iperf sama dengan langkah ke-empat namun nilai MTU pada masing-masing perangkat diberikan nilai sebesar 150, 300, 500, 1000, 1500 Bytes, sebelum dan sesudah dilakukan optimasi.

4.4.1.2 Hasil Simulasi TCP dengan Iperf

Dari simulasi didapatkan hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi.

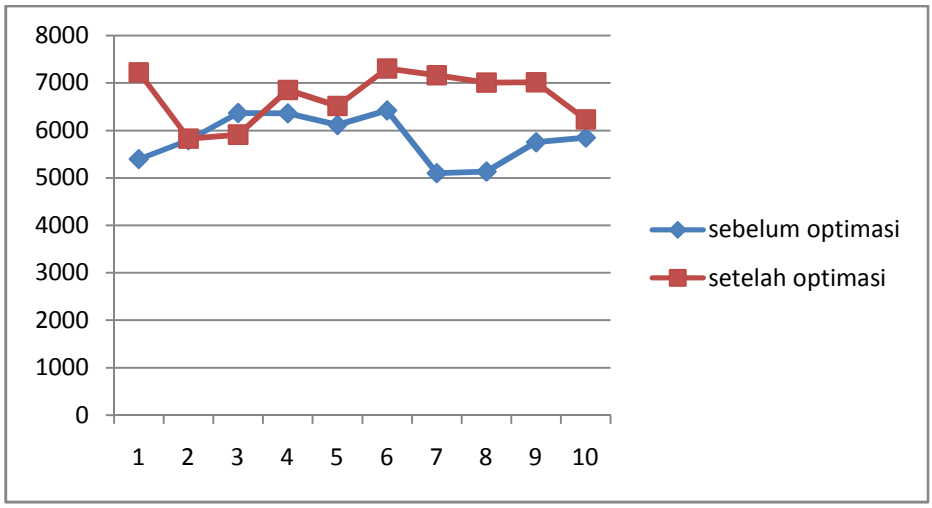
iterasi	Througput
1	5394
2	5792
3	6368
4	6359
5	6115
6	6420
7	5099
8	5133
9	5751
10	5847
average	5827,80

Tabel 4. 1 Hasil rata-rata simulasi Iperf sebelum Optimasi

iterasi	Througput
1	7220
2	5825
3	5908
4	6848
5	6511
6	7301
7	7158
8	7005
9	7013
10	6229
average	6701,80

Tabel 4. 2 Hasil rata-rata simulasi Iperf setelah Optimasi

Dari hasil statistik, dapat kita lihat bahwa performa untuk test paket TCP sangat baik. Terlihat peningkatan performa untuk pengiriman data setelah dilakukannya optimasi. Peningkatan throughput yang didapat ± 874 KB atau sekitar 15% pada saat PC router dioptimasi. Dari hasil diatas dapat dibuatkan diagram sebagai berikut.



Gambar 4.15 grafik perbandingan simulasi TCP dengan Iperf

4.4.1.3 Hasil Simulasi MTU dengan Iperf

Dari simulasi didapatkan hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi.

Sebelum Optimasi mtu 1500	
no	Kb/sec
1	5394
2	5792
3	6368
4	6359
5	6115
6	6420
7	5099
8	5133
9	5751
10	5847
	5827,80

Setelah Optimasi mtu 1500	
no	Kb/sec
1	7220
2	5825
3	5908
4	6848
5	6511
6	7301
7	7158
8	7005
9	7013
10	6229
	6701,80

mtu 1000	
no	Kb/sec
1	5527
2	5613
3	5572
4	5317
5	4619
6	5567
7	5479
8	5634
9	5544
10	5716
	5458,80

mtu 1000	
no	Kb/sec
1	5158
2	5256
3	5277
4	4939
5	4027
6	5247
7	5243
8	5162
9	5286
10	5229
	5082,40

mtu 500	
no	Kb/sec
1	2978
2	2879
3	3382
4	3735
5	3517
6	4413
7	2295
8	3125
9	3554
10	3247
	3312,50

mtu 500	
no	Kb/sec
1	4562
2	3558
3	4519
4	3532
5	4010
6	3743
7	4122
8	3600
9	4677
10	4257
	4058,00

mtu 300	
no	Kb/sec
1	2556
2	2724
3	2982
4	2614
5	2435
6	2417
7	3072
8	2279
9	2661
10	2348
	2608,80

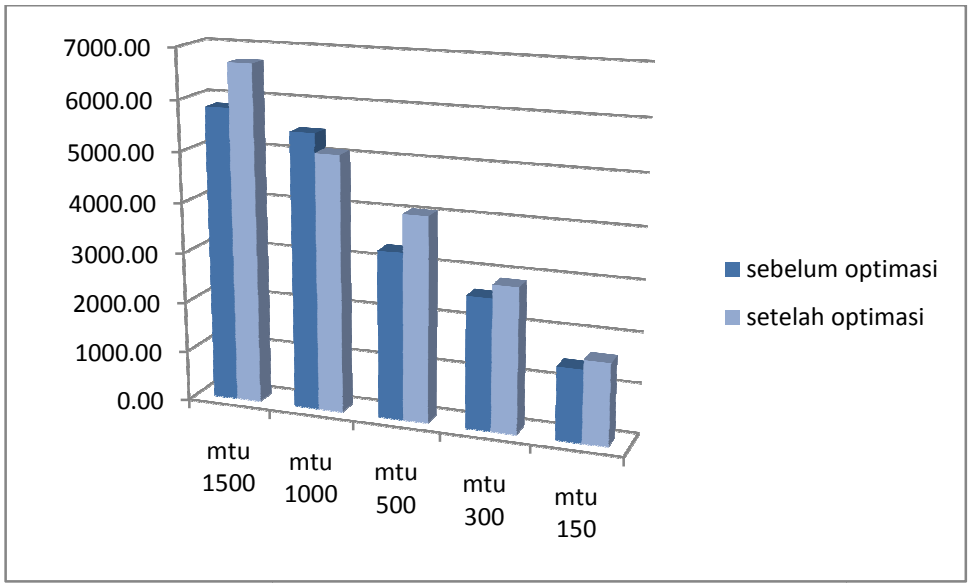
mtu 300	
no	Kb/sec
1	2712
2	3034
3	2694
4	2129
5	3243
6	3126
7	3109
8	3164
9	2756
10	2746
	2871,30

mtu 150	
no	Kb/sec
1	1220
2	1496
3	1336
4	1594
5	1591
6	1604
7	1283
8	1498
9	1223
10	1526
	1437,10

mtu 150	
no	Kb/sec
1	1227
2	1676
3	1617
4	1625
5	1689
6	1699
7	1617
8	1698
9	1661
10	1703
	1621,20

Tabel 4.3 perbandingan simulasi MTU pada Iperf

Dari data diatas didapatkan hasil bahaw perubahan MTU pada paket TCP ternyata merubah juga hasil througput yang didapat. Semakin kecil MTU, maka semakin kecil juga througput yang didapat. Ini menandakan semakin kecil MTU, maka semakin besar pula paket ACK (Acknowledge) yang dikeluarkan oleh TCP. Berikut diagram perbandingan antara sebelum dan sesudah optimasi.



Gambar 4.16 grafik perbandingan simulasi MTU pada Iperf

4.4.2 Simulasi menggunakan FTP

Simulasi berikutnya dilakukan dengan menggunakan aplikasi FTP seperti yang telah dijelaskan pada tahap perancangan. Simulasi FTP dilakukan dengan men-download file dengan ukuran dan kuantitas tertentu. Ada dua type file yang akan diuji, yaitu ;

1. Sebuah file *.avi yang berukuran 19.2MB
2. Beberapa tipe file yang diwakili oleh file *.txt, *.jpg, *.pdf, *.zip dan *.html yang dikumpulkan dalam satu folder berukuran 15,2MB. Untuk masing-masing file berjumlah satu buah.

Tujuan dari simulasi ini adalah untuk menggambarkan apakah type, besar dan kuantitas dari file mempengaruhi transfer data pada PC router.

4.4.2.1 Langkah-langkah Simulasi FTP

Pada simulasi FTP ada beberapa hal yang harus dipersiapkan sebelum dilakukan simulasi.

1. Install dan Aktifkan FTP server pada host pertama
2. Jalankan FTP client pada host kedua
3. Aktifkan IP routing pada PC router
4. Download file dengan perintah GET dan MGET pada FTP client dengan type Binary.
5. Pengambilan data dilakukan sebanyak 10 kali pengujian untuk masing-masing file yang di-download, pada saat sebelum dan sesudah dilakukannya optimasi.
6. Khusus untuk optimasi MTU, simulasi menggunakan FTP sama dengan langkah diatas namun nilai MTU pada masing-masing perangkat diberikan nilai sebesar 150, 300, 500, 1000, 1500 Bytes, sebelum dan sesudah dilakukan optimasi

4.4.2.2 Hasil Simulasi FTP pada sebuah Tipe File AVI

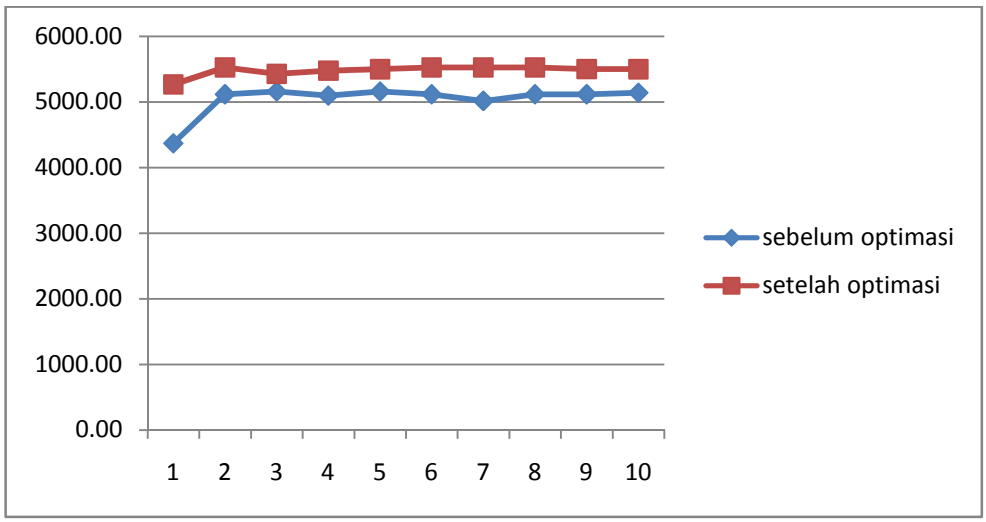
Dari simulasi didapatkan hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi.

no	Kb/sec
1	4369,49
2	5118,28
3	5160,07
4	5098,30
5	5160,07
6	5118,28
7	5016,07
8	5118,28
9	5118,28
10	5139,76
	5041,69

no	Kb/sec
1	5268,29
2	5523,76
3	5428,87
4	5475,90
5	5498,96
6	5523,76
7	5523,76
8	5523,76
9	5500,50
10	5498,96
	5476,65

Tabel 4. 4 Hasil rata-rata simulasi FTP untuk sebuah file type AVI

Pada tabel dapat kita lihat bahwa terjadi peningkatan performa transfer data pada FTP. Peningkatan dengan menggunakan PC router yang sama, menghasilkan peningkatan througput sebesar ± 435KB. Dari tabel diatas dapat kita buatkan grafik analisa seperti berikut.



Gambar 4.17 grafik perbandingan simulasi TCP pada FTP pada sebuah tipe file AVI

4.4.2.3 Hasil Simulasi FTP pada Tipe File yang Bervariasi

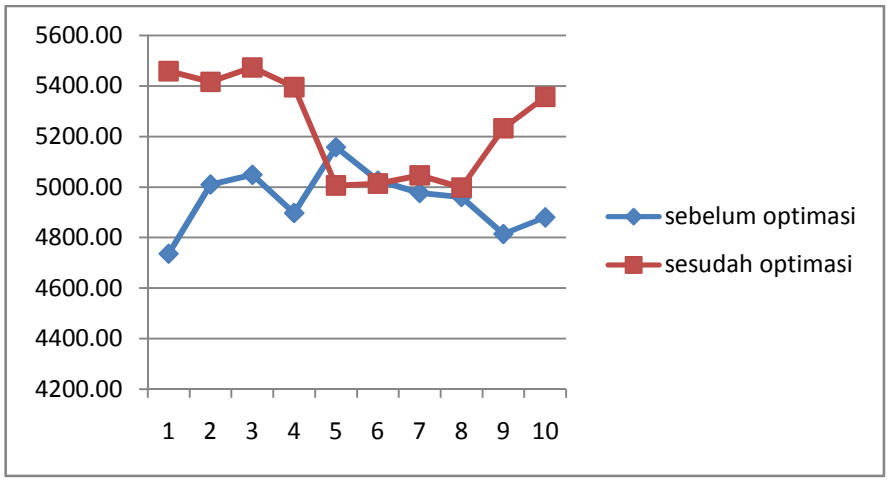
Dari simulasi didapatkan hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi.

no	Kb/sec
1	4735,56
2	5009,34
3	5047,98
4	4896,57
5	5157,10
6	5024,94
7	4976,47
8	4959,39
9	4813,59
10	4879,33
	4950,03

no	Kb/sec
1	5458,2
2	5415,962
3	5472,324
4	5394,884
5	5005,794
6	5013,678
7	5046,082
8	4997,998
9	5231,59
10	5355,85
	5239,24

Tabel 4. 5 Hasil rata-rata simulasi FTP untuk tipe file bervariasi

Untuk tipe file yang bervariasi, pada tabel terlihat peningkatan throughput sebesar ±289 KB. Hal ini membuktikan bahwa kuantitas dan tipe file sangat mempengaruhi hasil transfer data. Pada file pengambilan file AVI menggunakan FTP, peningkatan throughput terjadi ± 435 KB, sedangkan pada file yang bervariasi peningkatan througput hanya terjadi ± 289 KB.



Gambar 4.18 grafik perbandingan simulasi TCP pada FTP pada tipe file yang bervariasi

4.4.2.4 Hasil Simulasi MTU dengan FTP pada sebuah File AVI

Dari simulasi MTU didapatkan hasil pengujian rata-rata dari 10 iterasi dengan besar MTU 1500, 1000, 500, 300 dan 150 pada saat sebelum dan sesudah dilakukannya optimasi.

mtu 1500	
no	Kb/sec
1	4369,49
2	5118,28
3	5160,07
4	5098,30
5	5160,07
6	5118,28
7	5016,07
8	5118,28
9	5118,28
10	5139,76
	5041,69

mtu 1500	
no	Kb/sec
1	5268,29
2	5523,76
3	5428,87
4	5475,90
5	5498,96
6	5523,76
7	5523,76
8	5523,76
9	5500,50
10	5498,96
	5476,65

mtu 1000	
no	Kb/sec
1	4785,73
2	4956,43
3	5056,20
4	4956,43
5	5016,07
6	4995,61
7	5016,07
8	4975,31
9	5016,07
10	4995,61
	4976,95

mtu 1000	
no	Kb/sec
1	4879,90
2	5016,07
3	5077,17
4	5118,28
5	5077,17
6	5056,20
7	5096,98
8	5077,17
9	5075,85
10	5096,98
	5057,18

mtu 500	
no	Kb/sec
1	3918,56
2	3847,00
3	3822,98
4	3834,95
5	3666,31
6	3893,64
7	3788,24
8	3710,05
9	3968,56
10	4058,16
	3850,85

mtu 500	
no	Kb/sec
1	3743,37
2	3666,31
3	3687,71
4	3666,31
5	3765,67
6	3634,35
7	3532,14
8	3788,97
9	3788,24
10	3721,33
	3699,44

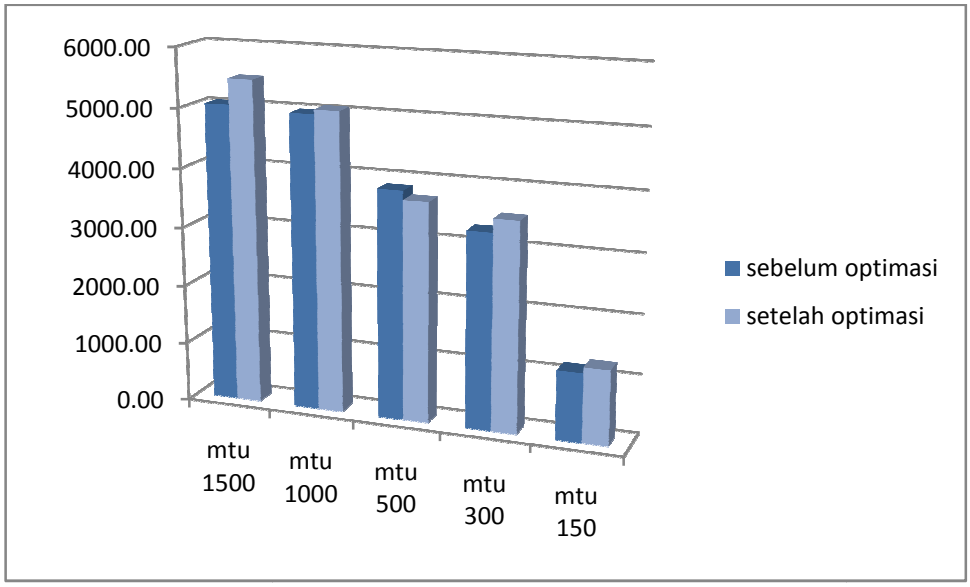
mtu 300	
no	Kb/sec
1	3282,42
2	3464,07
3	3231,53
4	3207,20
5	3407,44
6	3150,98
7	3425,91
8	3389,17
9	3198,82
10	3223,56
	3298,11

mtu 300	
no	Kb/sec
1	3512,51
2	3522,61
3	3512,51
4	3522,61
5	3522,61
6	3521,98
7	3532,78
8	3521,98
9	3542,36
10	3532,14
	3524,41

mtu 150	
no	Kb/sec
1	1187,44
2	1129,66
3	1169,72
4	1148,26
5	992,84
6	1197,68
7	1197,68
8	1152,52
9	1158,92
10	1287,39
	1162,21

mtu 150	
no	Kb/sec
1	1204,52
2	1255,22
3	1217,47
4	1338,21
5	1379,48
6	1294,11
7	1296,68
8	1212,72
9	1151,50
10	1279,49
	1262,94

Tabel 4. 5 Hasil rata-rata simulasi MTU pada FTP untuk tipe file AVI



Gambar 4.18 grafik perbandingan simulasi MTU pada FTP untuk tipe file yang AVI Sebelum dan Setelah Optimasi

Terlihat pada grafik menunjukan bahwa nilai MTU yang dipakai mempengaruhi hasil throughput pada PC router. Terjadi pengurangan hasil throughput jika MTU diset semakin kecil, karena pada pengiriman paket AVI yang besar semakin efisien jika MTU diset semaksimal mungkin.

4.4.2.5 Hasil Simulasi MTU dengan FTP pada Tipe File yang bervariasi

mtu 1500	
no	Kb/sec
1	4735,56
2	5009,34
3	5047,98
4	4896,57
5	5157,10
6	5024,94
7	4976,47
8	4959,39
9	4813,59
10	4879,33
	4950,03

mtu 1500	
no	Kb/sec
1	5458,2
2	5415,962
3	5472,324
4	5394,884
5	5005,794
6	5013,678
7	5046,082
8	4997,998
9	5231,59
10	5355,85
	5239,24

mtu 1000	
no	Kb/sec
1	4588,12
2	4805,82
3	4976,05
4	4988,65
5	5041,85
6	4894,17
7	4772,99
8	4756,08
9	4878,99
10	4857,98
	4856,07

mtu 1000	
no	Kb/sec
1	4917,414
2	4959,386
3	4978,872
4	4961,732
5	4962,696
6	4926,048
7	4876,116
8	4863,986
9	4835,97
10	4960,852
	4924,31

mtu 500	
no	Kb/sec
1	3576,46
2	3647,38
3	3439,14
4	3590,18
5	3778,23
6	3661,60
7	3584,51
8	3575,30
9	3577,41
10	3748,88
	3617,91

mtu 500	
no	Kb/sec
1	4171,882
2	4093,746
3	4250,404
4	4252,178
5	4186,43
6	4250,404
7	4396,698
8	4089,73
9	4123,932
10	4065,71
	4188,11

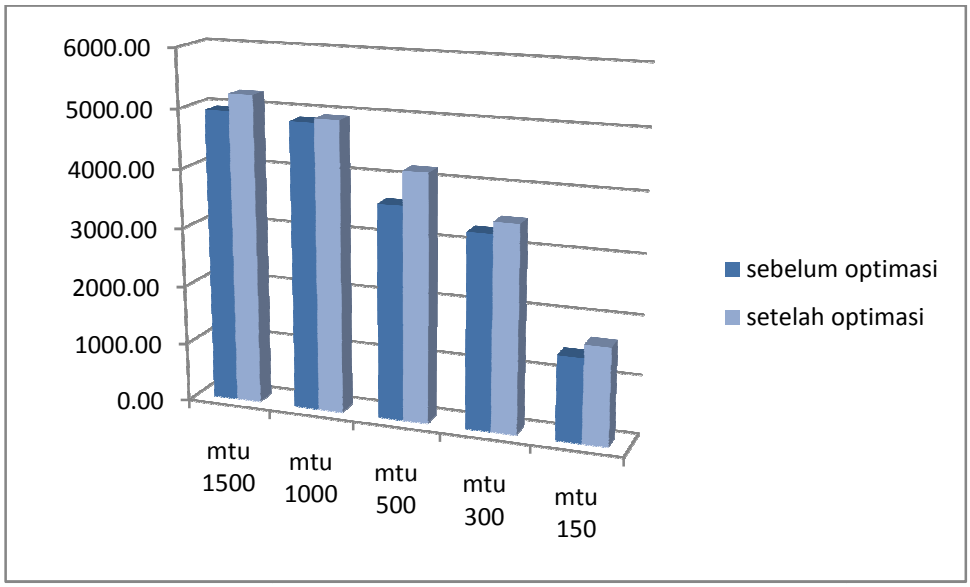
mtu 300	
no	Kb/sec
1	3290,58
2	3209,31
3	3301,25
4	3237,36
5	3385,72
6	3320,30
7	3343,56
8	3345,97
9	3407,56
10	3054,10
	3289,57

mtu 300	
no	Kb/sec
1	3479,01
2	3429,436
3	3468,642
4	3474,314
5	3478,904
6	3474,002
7	3542,7
8	3494,468
9	3507,226
10	3517,152
	3486,59

mtu 150	
no	Kb/sec
1	1431,57
2	1413,29
3	1269,24
4	1453,89
5	1422,18
6	1340,84
7	1464,23
8	1421,94
9	1432,54
10	1606,05
	1425,58

mtu 150	
no	Kb/sec
1	1544,812
2	1683,13
3	1657,312
4	1618,068
5	1655,882
6	1615,26
7	1682,842
8	1667,27
9	1660,31
10	1686,494
	1647,14

Tabel 4. 5 Hasil rata-rata simulasi MTU pada FTP untuk tipe file bervariasi



Gambar 4.19 grafik perbandingan simulasi MTU pada FTP untuk tipe file yang bervariasi

4.4.2.6 Rangkuman Hasil Simulasi FTP

Dari simulasi MTU didapatkan sebuah rangkuman, bahwa pada pengiriman tipe file AVI dengan tipe file yang bervariasi menghasilkan perbedaan peningkatan hasil throughput. Pada pengujian menggunakan tipe file AVI, peningkatan throughput terjadi $\pm$ 435 KB atau sekitar 8,6%. Sedangkan pada pengujian menggunakan tipe file yang bervariasi terjadi peningkatan througput sebesar $\pm$ 289 KB atau sekitar 5,8%. Didapatkan juga hasil bahwa setiap

pengurangan MTU pada tiap pengujian FTP menghasilkan pengurangan throughput, hasil dapat lebih buruk untuk PC router yang telah dioptimasi sekalipun, namun masih pada nilai MTU tertentu saja.

4.4.3 Simulasi pada Protokol UDP dengan Iperf

Pada simulasi berikut digunakan protokol UDP yang bersifat connectionless. Aplikasi yang digunakan adalah Iperf. Pada simulasi ini ukuran data yang akan ditransfer sebesar 1 Mbit/sec.

4.4.3.1 Langkah-langkah Simulasi UDP dengan Iperf

Seperti kita ketahui simulasi menggunakan Qcheck bersifat end-to-end point. Pada simulasi ini ada beberapa hal yang harus dipersiapkan sebelum dilakukan simulasi.

1. Mengkonfigurasi Iperf sebagai server pada host pertama.
2. Mengkonfigurasi Iperf sebagai client pada host kedua.
3. Aktifkan IP routing pada PC router
4. Mencatat transfer rate yang dihasilkan pada transfer data menggunakan Iperf selama 10 second dengan besar bandwith 1 Mbits/sec, dengan minimal pengambilan data 10 iterasi pada saat sebelum dan sesudah dioptimasi.
5. Khusus untuk optimasi MTU, simulasi menggunakan Iperf sama dengan langkah ke-empat namun nilai MTU pada masing-masing perangkat diberikan nilai sebesar 150, 300, 500, 1000, 1500 Bytes, sebelum dan sesudah dilakukan optimasi.

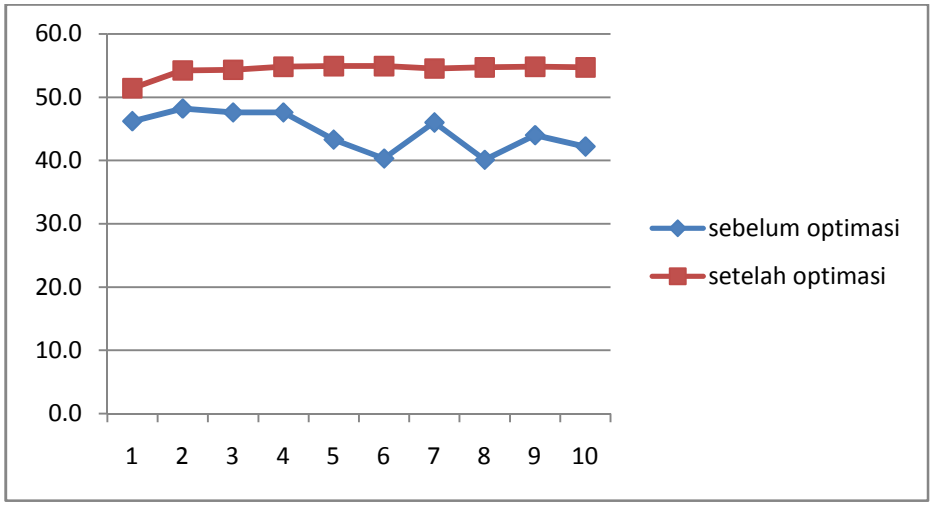
4.4.3.2 Hasil Simulasi UDP dengan Iperf

Dari simulasi didapatkan hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi.

Througput PC router sebelum dioptimasi	
no	Mbit/sec
1	46,2
2	48,2
3	47,6
4	47,6
5	43,3
6	40,3
7	46,0
8	40,1
9	44,0
10	42,2
	44,6

mtu 1500 Througput PC router setelah dioptimasi	
no	Mbit/sec
1	51,4
2	54,2
3	54,3
4	54,8
5	54,9
6	54,9
7	54,5
8	54,7
9	54,8
10	54,7
	54,3

Tabel 4. 5 Hasil rata-rata simulasi UDP pada Iperf



Gambar 4.20 grafik perbandingan simulasi UDP pada Iperf

Pada grafik terlihat peningkatan performa throughput dari 44.6 Mbits/sec menjadi 54.3 Mbits/sec. Peningkatan throughput terjadi sebesar 9,7 Mbits/sec atau sekitar 21.75%. Hal ini membuktikan bahwa pada protokol UDP, optimasi yang dilakukan pada PC router dapat meningkatkan performa throughput sebesar 21.75%.

4.4.3.3 Hasil Simulasi MTU pada protokol UDP dengan Iperf

Dari simulasi didapatkan hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi.

mtu 1500	
no	Mbit/sec
1	46,2
2	48,2
3	47,6
4	47,6
5	43,3
6	40,3
7	46,0
8	40,1
9	44,0
10	42,2
	44,6

mtu 1500	
no	Mbit/sec
1	51,4
2	54,2
3	54,3
4	54,8
5	54,9
6	54,9
7	54,5
8	54,7
9	54,8
10	54,7
	54,3

mtu 1000	
no	Mbit/sec
1	45,1
2	44,0
3	43,7
4	45,1
5	41,7
6	44,1
7	44,2
8	42,4
9	42,6
10	45,9
	43,9

mtu 1000	
no	Mbit/sec
1	48,5
2	48,3
3	48,5
4	48,1
5	48,6
6	47,6
7	48,6
8	48,2
9	48,5
10	47,3
	48,2

mtu 500	
no	Mbit/sec
1	33,1
2	34,2
3	32,7
4	33,6
5	31,2
6	33,1
7	33,8
8	33,3
9	33,7
10	31,6
	33,0

mtu 500	
no	Mbit/sec
1	33,5
2	34,5
3	34,6
4	33,6
5	34,7
6	34,6
7	32,7
8	26,4
9	34,4
10	34,5
	33,4

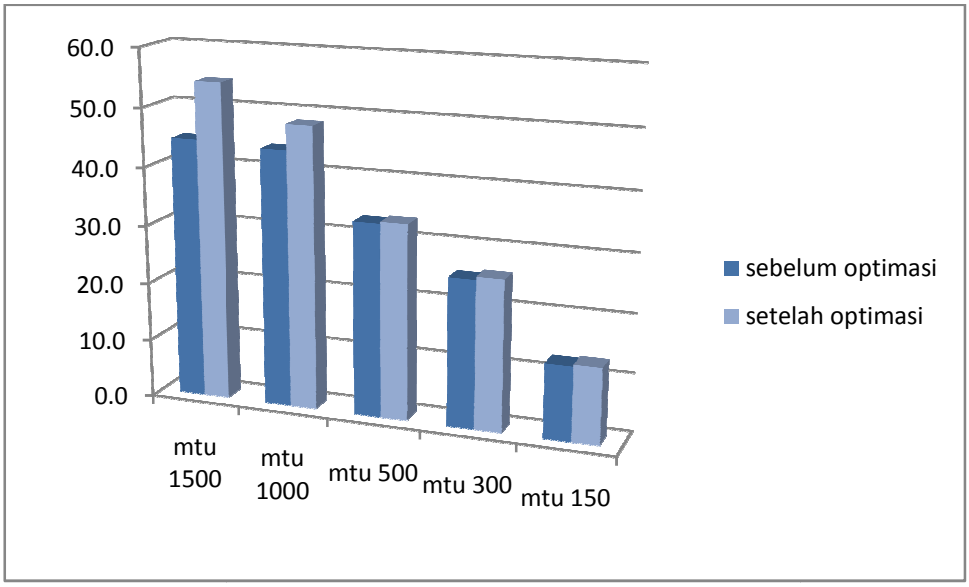
mtu 300	
no	Mbit/sec
1	25,2
2	24,6
3	24,8
4	25,5
5	24,9
6	24,8
7	25,9
8	25,5
9	25,6
10	25,5
	25,2

mtu 300	
no	Mbit/sec
1	25,9
2	26,4
3	25,3
4	26,3
5	25,4
6	25,2
7	25,9
8	25,3
9	26,3
10	25,6
	25,8

mtu 150	
no	Mbit/sec
1	13,0
2	13,2
3	13,1
4	13,1
5	12,9
6	13,1
7	12,4
8	11,8
9	12,0
10	12,1
	12,7

mtu 150	
no	Mbit/sec
1	13,1
2	13,3
3	13,4
4	10,3
5	13,3
6	13,4
7	13,3
8	13,3
9	13,3
10	13,3
	13,0

Tabel 4. 5 Hasil rata-rata simulasi MTU UDP pada Iperf



Gambar 4.21 grafik perbandingan simulasi MTU UDP pada Iperf

Pada tabel dan grafik diatas, peningkatan performa terjadi untuk semua nilai MTU pada PC router setelah dioptimasi. Peningkatan untuk MTU 500 kebawah terjadi lebih sedikit, hanya sekitar 1,2 - 2,3%. Untuk paket UDP dengan nilai MTU 1000 dan 1500 didapat kenaikan sekitar 8.9 dan 21,7%.

4.4.4 Simulasi Streaming dengan Qcheck

Streaming adalah salah satu apliklasi real time yang dapat menggunakan protokol UDP dalam mengirimkan data. Simulasi streaming yang akan dilakukan menggunakan aplikasi Qcheck dari Ixia. Data yang digunakan pada simulasi streaming ini bukan merupakan tipe data video atau multimedia yang biasa digunakan untuk streaming tetapi merupakan data streaming yang di-*generate* oleh aplikasi Qcheck.

4.4.4.1 Langkah-langkah Simulasi Streaming

Simulasi streaming ini dilakukan pada protokol UDP. Qcheck diinstall pada dua host, namun yang di aktifkan hanya satu host saja. Isikan IP addres host yang akan dilewatkan oleh data. Pada menu pilihan ambil protokol UDP lalu streaming. Kemudian tentukan besar data rate untuk streaming dengan nilai 1000 Kbps atau 1 Mbps dengan durasi setiap streaming selama 5 detik dan dilakukan sebanyak 10 kali iterasi.

4.4.4.2 Hasil Simulasi MTU dengan Qcheck untuk Streaming

Khusus untuk simulasi UDP streaming dengan Qcheck hasil simulasi dirangkum sekaligus dengan hasil simulasi MTU. Mengingat simulasi ini sangat sedikit sekali perbandingannya. Hasil pengujian rata-rata dari 10 iterasi untuk PC router sebelum dan sesudah dilakukannya optimasi dapat dilihat pada tabel berikut.

mtu 1500	
no	Kbps
1	1000,436
2	1000,603
3	1000,436
4	1000,436
5	1000,436
6	1000,603
7	1000,603
8	1000,436
9	1000,603
10	1000,436
	1000,503

mtu 1500	
no	Kbps
1	1000,436
2	1000,603
3	1000,603
4	1000,603
5	1000,603
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,603
	1000,520

mtu 1000	
no	Kbps
1	1000,436
2	995,792
3	1000,436
4	1000,603
5	1000,603
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,603
	1000,022

mtu 1000	
no	Kbps
1	1000,603
2	1000,436
3	1000,436
4	1000,436
5	1000,436
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,436
	1000,453

mtu 500	
no	Kbps
1	1000,436
2	1000,436
3	1000,436
4	1000,436
5	1000,603
6	1000,436
7	1000,770
8	1000,436
9	1000,436
10	1000,436
	1000,486

mtu 500	
no	Kbps
1	1000,269
2	1000,603
3	1000,436
4	1000,436
5	1000,436
6	1000,770
7	1000,603
8	1000,436
9	1000,436
10	1000,436
	1000,486

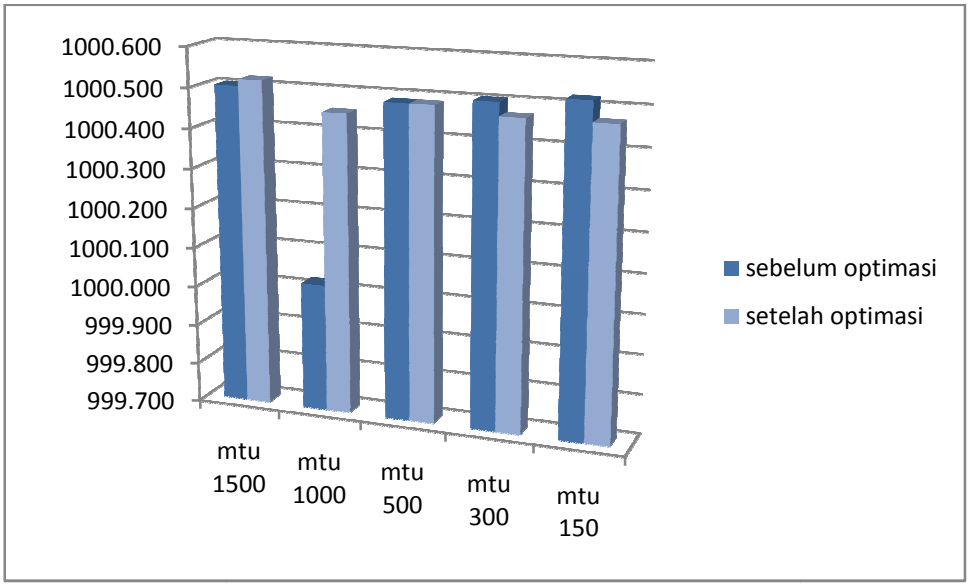
mtu 300	
no	Kbps
1	1000,436
2	1000,436
3	1000,603
4	1000,436
5	1000,770
6	1000,436
7	1000,603
8	1000,436
9	1000,436
10	1000,436
	1000,503

mtu 300	
no	Kbps
1	1000,436
2	1000,436
3	1000,770
4	1000,269
5	1000,603
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,436
	1000,469

mtu 150	
no	Kbps
1	1000,603
2	1000,603
3	1000,436
4	1000,436
5	1000,436
6	1000,436
7	1000,436
8	1000,603
9	1000,436
10	1000,770
	1000,520

mtu 150	
no	Kbps
1	1000,603
2	1000,436
3	1000,603
4	1000,436
5	1000,436
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,436
	1000,469

Tabel 4. 5 Hasil rata-rata simulasi MTU UDP Streaming pada Qcheck



Gambar 4.21 grafik perbandingan simulasi MTU UDP Streaming pada Qcheck

Dari hasil simulasi *streaming* diperoleh bahwa kenaikan *throughput* antara nilai MTU 1500 dengan MTU 150 pada MTU 1500 yang telah dioptimasi dan MTU 150 pada PC router yang belum teroptimasi. Pada dasarnya jika dilihat pada nilai-nilai yang tertera pada masing-masing MTU, nilai throughput rata-rata yang sering terlihat antara 1000,436 , 1000,603 , dan 1000,770. Hal ini membuktikan bahwa kenaikan throughput pada saat streaming tidak terlalu berarti. Sekitar 0,017 Kbps. Namun pada nilai MTU 1000 terlihat bahwa packet lost yang ada sangat besar, sehingga penurunan throughput yang didapat juga besar. Hasil simulasi *streaming* ini tidak menunjukkan suatu pola yang tidak tetap pada tiap nilai MTU. Pada MTU 1000 dan 1500 terdapat kenaikan pada PC router yang telah dioptimasi, kemudian pada nilai MTU 500 didapatkan hasil yang sama, namun pada nilai MTU 150 dan 300 kenaikan throughput justru terjadi pada saat PC belum dioptimasi.

4.5 Perbandingan Transfer Data PC Router dengan Linux Live-CD Router.

Pada pengujian kali ini kita akan menguji Linux Live-CD Router dengan PC router dengan sistem operasi Red Hat 9.0. Pengujian meliputi pada paket TCP saja. Pengujian akan dilakukan sebagai perbandingan dengan PC router yang telah kita optimasi. Sesuai namanya Linux Live-CD Router merupakan Linux yang

dapat berjalan langsung pada kepingan CD, tanpa terlebih dahulu di-install kedalam hard drive. Versi Linux Live-CD Router ini merupakan versi gratisan (*free*) dengan fitur terbatas, namun tetap dapat menjalankan routing pada PC router. Berikut fitur Linux Live-CD Router Pro secara keseluruhan;

1. Share koneksi internet broadband
2. Support untuk xDSL, Cablemodem, Fixed IP, Dial-Up dan WIFI
3. Termasuk juga Firewall Shorewall
4. Web Administration
5. Koneksi Remote SSH
6. DNS Chace untuk meningkatkan performa internet
7. SNMP remote monitoring dilengkapi graphical statistic
8. Compatble dengan Windows dan Mac Networks
9. Tidak memerlukan hardisk dan instalasi
10. Slackware Distribution Base
11. Kernel 2.4.31

Namun untuk versi gratisan terbatas pada routing dan koneksi SSH saja. Fitur lainnya hanya didapat jika kita membeli versi yang Linux Live-CD Router Pro dengan harga USD 55,00.

4.5.1.1 Langkah-langkah Simulasi menggunakan Linux Live-CD Router.

Pada simulasi ini kita yang kita ujikan pada protokol TCP dan UDP adalah dengan menggunakan aplikasi Iperf. Iperf kita gunakan karena secara dukungannya terhadap kedua protokol diatas. Langkah-langkah menggunakan Iperf sama dengan langkah-langkah penggunaan pada pengujian diatas. Khusus pada pengujian ini, tidak diikuti dengan pengambilan data pada nilai MTU. Nilai MTU yang ditetapkan disini adalah nilai MTU default atau 1500 bytes.

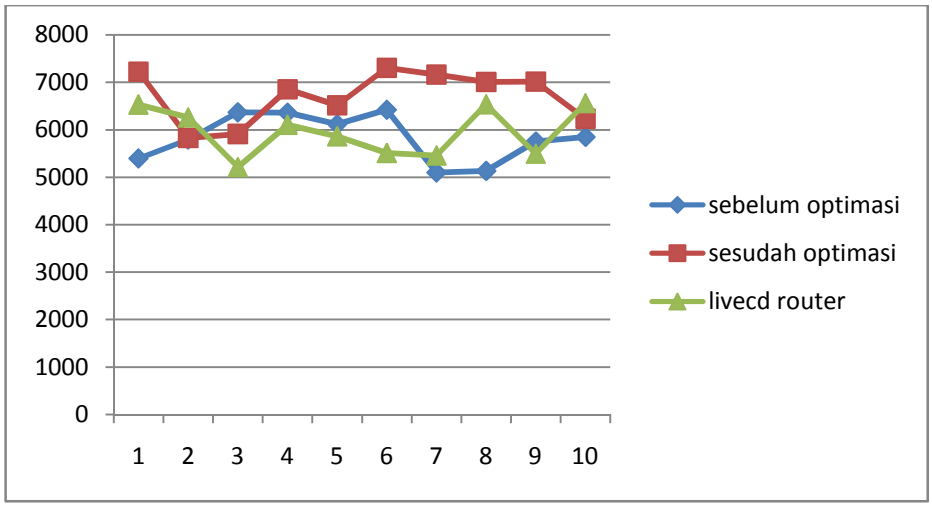
4.5.1.2 Hasil Simulasi TCP dengan Iperf

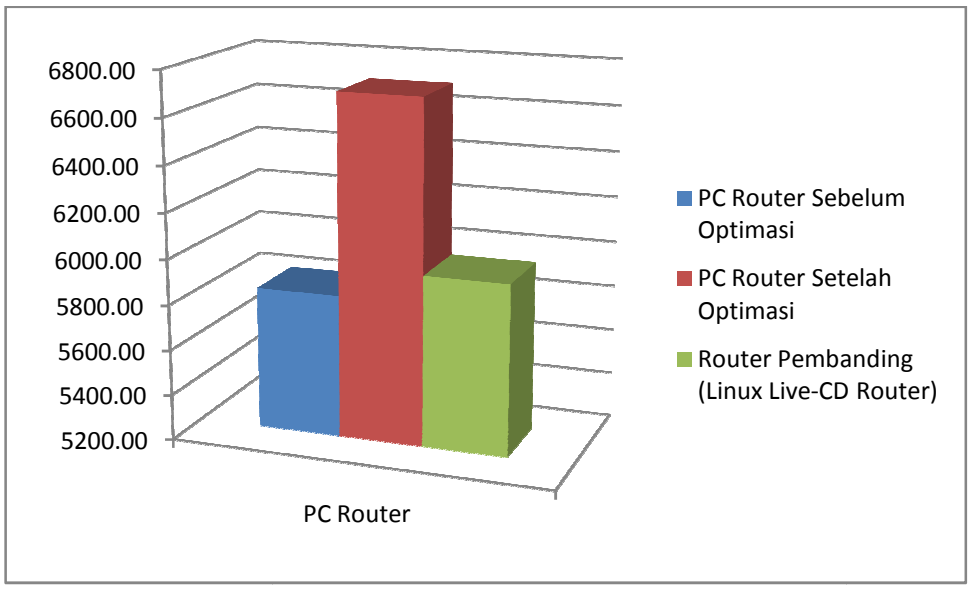
mtu 1500	
no	Kb/sec
1	5394
2	5792
3	6368
4	6359
5	6115
6	6420
7	5099
8	5133
9	5751
10	5847
	5827,80

mtu 1500	
no	Kb/sec
1	7220
2	5825
3	5908
4	6848
5	6511
6	7301
7	7158
8	7005
9	7013
10	6229
	6701,80

mtu 1500	
no	Kb/sec
1	6527
2	6257
3	5215
4	6104
5	5860
6	5511
7	5459
8	6535
9	5495
10	6557
	5952

Tabel 4. 5 Hasil rata-rata simulasi MTU UDP Streaming pada Qcheck





4.5.1.3 Hasil Simulasi UDP dengan Iperf

PC Router Sebelum Optimasi		PC Router Setelah Optimasi		Router Pembanding (Linux Live-CD Router)	
no	Mbit/sec	no	Mbit/sec	no	Mbit/sec
1	46,2	1	51,4	1	44,2
2	48,2	2	54,2	2	44,2
3	47,6	3	54,3	3	43,0
4	47,6	4	54,8	4	43,1
5	43,3	5	54,9	5	44,4
6	40,3	6	54,9	6	44,4
7	46,0	7	54,5	7	43,3
8	40,1	8	54,7	8	43,9
9	44,0	9	54,8	9	43,3
10	42,2	10	54,7	10	42,1
	44,6		54,3		43,6

Tabel 4. 5 Hasil rata-rata simulasi MTU UDP Streaming pada Qcheck

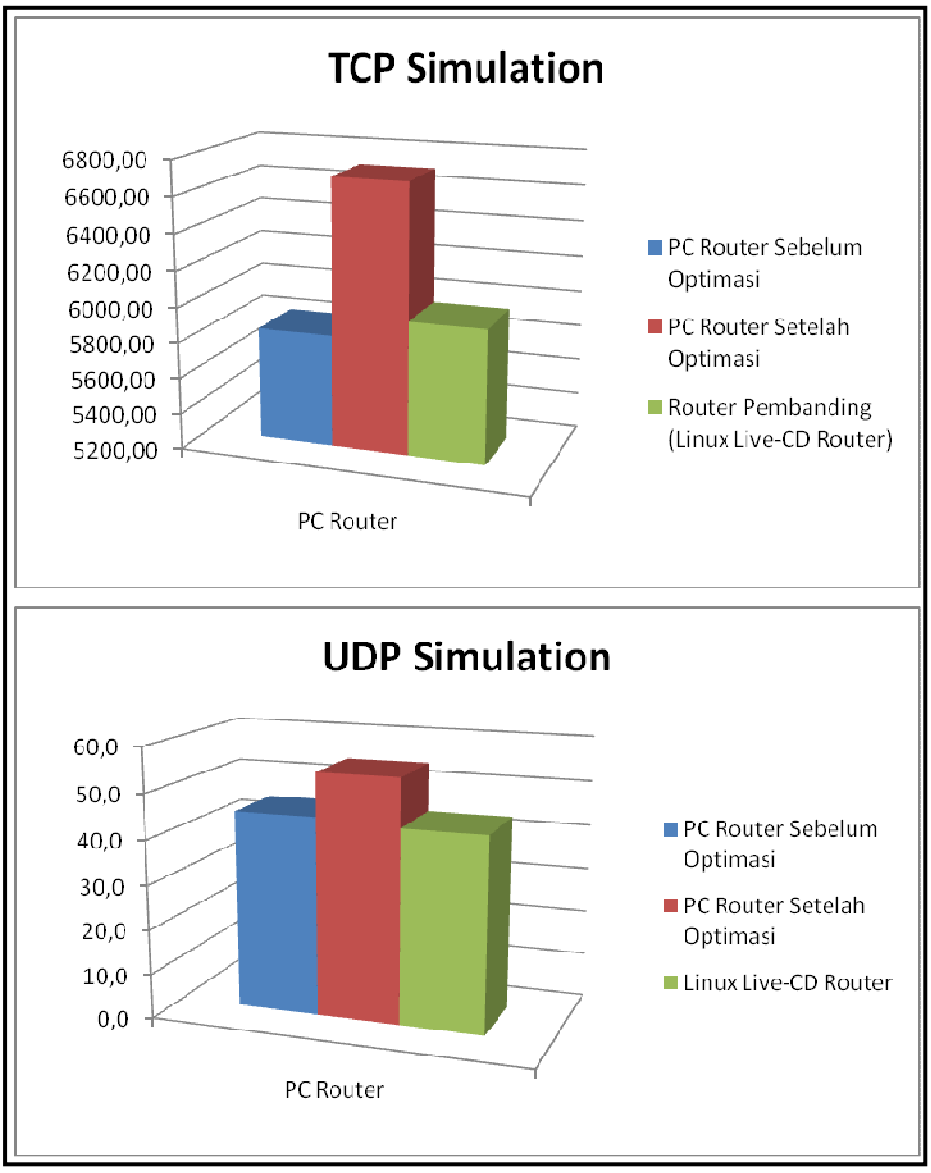
BAB V

PENUTUP

5.1 Kesimpulan

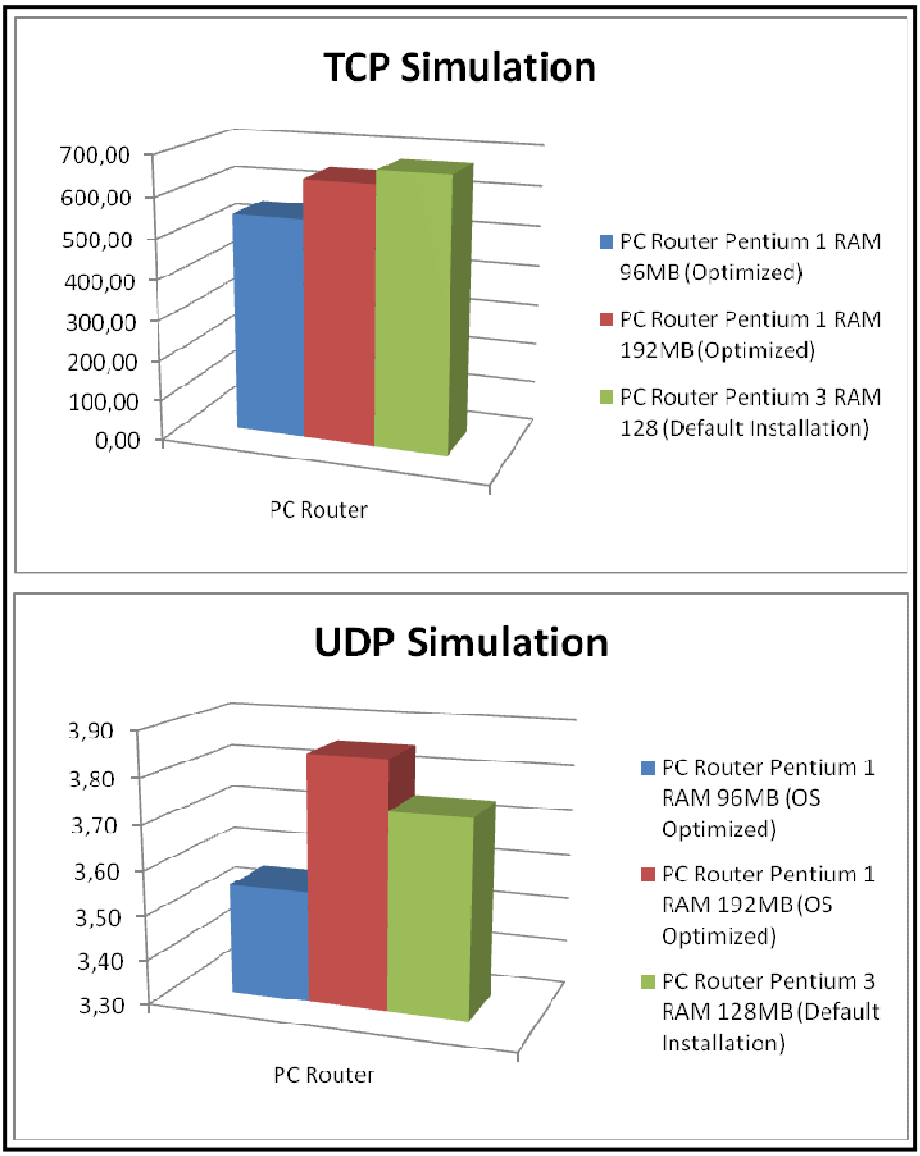
Setelah menganalisa dan menguji melalui simulasi transfer data pada PC *router*, maka dapat diambil kesimpulan yaitu:

1. *Compile kernel* dan *upgrade kernel* yang dilakukan merupakan cara terbaik untuk mendapatkan sistem yang kita butuhkan. Konfigurasi yang tepat merupakan cara untuk mendapatkan hasil yang optimal.
2. Manajemen memori yang baik dapat kita lakukan dengan memilah-milah *service* yang kita butuhkan. Semakin banyak *service* yang dijalankan, maka semakin tinggi pula alokasi memori yang kita butuhkan.
3. Perubahan untuk nilai MTU (*Maximum Transfer Unit*) dari nilai *default* khususnya pada teknologi *ethernet* tidak menghasilkan peningkatan *throughput* pada saat *transfer* data dilakukan pada PC router. Sebaliknya, penurunan *throughput* dirasakan jika nilai MTU diset semakin kecil dari nilai *default*.
4. Optimasi optimum yang dilakukan pada PC *router* dengan sistem operasi *Red Hat 9.0* (*Shrike*), *kernel* 2.4.20 adalah dengan meng-*upgrade kernel* 2.4.32 dan mengoptimasi *service* yang dijalankan dengan nilai MTU *default*.
5. Faktor yang mempengaruhi performa transfer data pada PC router yang terlihat pada pengujian diatas adalah:
    a. Tipe file yang bervariasi.
    b. Protokol yang digunakan.
    c. Spesifikasi PC *router*.



Gambar 5.1 Grafik perbandingan performa throughput TCP dan UDP pada Linux Live-CD Router pada nilai MTU default.

6. Dari grafik diatas dapat disimpulkan peningkatan throughput pada *PC router* yang telah teroptimasi pada protokol TCP meningkat hingga 15% dan pada protokol UDP meningkat hingga 22%, pada saat dibandingkan dengan *PC router* yang belum dioptimasi.
7. Pada perbandingan dengan *Linux Live-CD Router* peningkatan performa *throughput* terjadi pada *PC router* yang telah dioptimasi dengan nilai peningkatan sebesar 12,6% pada protokol TCP dan 24,6 % pada protokol UDP.



Gambar 5.2 Grafik perbandingan performa throughput TCP dan UDP pada PC router dengan Spesifikasi Berbeda

8. Pada gambar 5.2 dapat diambil kesimpulan bahwa peningkatan *throughput* pada protokol TCP dapat dilakukan dengan meningkatkan *hardware* terutama pada *processor*. Peningkatan pada PC router *Pentium 1* yang telah di-*upgrade* alokasi *memory*-nya terjadi sebesar 17,4%. Sedangkan peningkatan *throughput* pada PC *router* yang menggunakan processor *Pentium III* terjadi sekitar 23,4%
9. *Throughput* yang dihasilkan pada protokol UDP pada PC *router* dengan alokasi *memory* yang telah ditingkatkan ternyata membawa dampak pada peningkatan *throughput* dibandingkan dengan pada PC *router* yang telah

ditingkatkan *processor*-nya. Dari hal diatas dapat disimpulkan, bahwa peningkatan pada alokasi *memory* pada protokol UDP dapat meningkatkan performa *throughput* pada PC *router* dari pada peningkatan pada *processor*. Peningkatan terjadi sebesar 8,5% pada PC *router* yang telah ditingkatkan alokasi *memory*-nya, sedangkan pada PC router *Pentium III* peningkatan *thorughput* terjadi hanya sebesar 5.4%.

5.2 Saran

Dalam simulasi transfer data pada PC *router* ini, penulis menyadari bahwa simulasi ini memiliki kekurangan serta kelemahan. Beberapa saran yang ditujukan untuk perbaikan yaitu:

1. Simulasi ini belum sepenuhnya diterapkan pada *traffic* jaringan sebenarnya. Diharapkan kedepan akan dapat diadakan penelitian menggunakan PC *router* pada *traffic* yang benar-benar sibuk.
2. Penggunaan *routing protokol* layak diterapkan disini, mengingat alokasi dari *resource* yang telah teroptimasi dapat dioptimalkan dengan penggunaan *routing* protokol untuk koneksi beberapa PC *router* lainnya.
3. Walaupun hasil pengujian pada PC *router* yang telah dioptimasi terjadi peningkatan performa dibandingkan dengan *Linux Live-CD Router*, namun fitur yang tersedia pada PC *router* yang telah dioptimasi sangat minim sekali dibandingkan dengan *Linux Live-CD Router*. Diharapkan kedepan diadakan peningkatan fitur pada PC *router*.
4. Perubahan nilai MTU (*Maximum Transfer Unit*) pada *Ethernet* ternyata menghasilkan penurunan throughput. Namun apakah pada *Gigabit Ethernet* didapatkan hasil yang sama? Untuk itu perlu kiranya diadakan penelitian serupa untuk PC *router* dengan *interface Gigabit Ethernet*.

DAFTAR PUSTAKA

Bouet, Daniel P. dan Marco Cesati. 2002. *Understanding The Linux Kernel, 2nd Edition*. O'Reilly

Inc, Global Knowledge. 1999. *Introduction to Wide Area Networking*. Stiefelmeyer International, Ltd.

Kirch, Olaf dan Terry Dawson. 2000. *Linux Network Administrator Guide*. O'Reilly

Kelompok Kerja 21–28 IKI-20230, Gabungan. 2003. Sistem Operasi: Bahan Kuliah IKI-20230. *GNU Free Documentation License versi 1.1*

Kroah, Greg dan Hartman. 2005. *Linux Kernel in A Nutshell*. O'Reilly

Lammle, Tod. 2006. *Introducing to Cisco Networking Technologies*. Willey Publishing.

Odom, Wendell. 2004. *Computer Networking FIRST-STEP*. Andi Yogyakarta.

Tanembaum, Andrew S. 1997. *Jaringan Komputer, Edisi Bahasa Indonesia dari Komputer Network Edisi III*. Jakarta. Prenhallindo.

Tayler, Chris. 2006. *Fedora Linux*. O'Reilly

Purbo, Ono W. 2001. *TCP/IP – Standar, desain, dan Implementasi*. Elex Media Komputindo.

Wijaya, Hendra. 2003. Belajar Sendiri *Cisco Router*. Elex Media Komputindo.

LAMPIRAN A

TABEL

A -1. Nilai Throughput yang didapat pada pengujian MTU pada protokol TCP dengan menggunakan Iperf.

Nilai Throughput PC Router Sebelum Dioptimasi mtu 1500	
no	Kb/sec
1	5394
2	5792
3	6368
4	6359
5	6115
6	6420
7	5099
8	5133
9	5751
10	5847
	5827,80

Nilai Throughput PC Router Setelah Dioptimasi mtu 1500	
no	Kb/sec
1	7220
2	5825
3	5908
4	6848
5	6511
6	7301
7	7158
8	7005
9	7013
10	6229
	6701,80

mtu 1000	
no	Kb/sec
1	5527
2	5613
3	5572
4	5317
5	4619
6	5567
7	5479
8	5634
9	5544
10	5716
	5458,80

mtu 1000	
no	Kb/sec
1	5158
2	5256
3	5277
4	4939
5	4027
6	5247
7	5243
8	5162
9	5286
10	5229
	5082,40

no	Kb/sec
1	2978
2	2879
3	3382
4	3735
5	3517
6	4413
7	2295
8	3125
9	3554
10	3247
	3312,50

no	Kb/sec
1	4562
2	3558
3	4519
4	3532
5	4010
6	3743
7	4122
8	3600
9	4677
10	4257
	4058,00

mtu 300	
no	Kb/sec
1	2556
2	2724
3	2982
4	2614
5	2435
6	2417
7	3072
8	2279
9	2661
10	2348
	2608,80

mtu 300	
no	Kb/sec
1	2712
2	3034
3	2694
4	2129
5	3243
6	3126
7	3109
8	3164
9	2756
10	2746
	2871,30

mtu 150	
no	Kb/sec
1	1220
2	1496
3	1336
4	1594
5	1591
6	1604
7	1283
8	1498
9	1223
10	1526
	1437,10

mtu 150	
no	Kb/sec
1	1227
2	1676
3	1617
4	1625
5	1689
6	1699
7	1617
8	1698
9	1661
10	1703
	1621,20

A -2. Nilai Throughput yang didapat pada pengujian MTU pada protokol TCP dengan menggunakan FTP untuk sebuah file AVI.

Nilai Throughput PC Router Sebelum Dioptimasi mtu 1500	
no	Kb/sec
1	4369,49
2	5118,28
3	5160,07
4	5098,30
5	5160,07
6	5118,28
7	5016,07
8	5118,28
9	5118,28
10	5139,76
	5041,69

Nilai Throughput PC Router Setelah Dioptimasi mtu 1500	
no	Kb/sec
1	5268,29
2	5523,76
3	5428,87
4	5475,90
5	5498,96
6	5523,76
7	5523,76
8	5523,76
9	5500,50
10	5498,96
	5476,65

mtu 1000	
no	Kb/sec
1	4785,73
2	4956,43
3	5056,20
4	4956,43
5	5016,07
6	4995,61
7	5016,07
8	4975,31
9	5016,07
10	4995,61
	4976,95

mtu 1000	
no	Kb/sec
1	4879,90
2	5016,07
3	5077,17
4	5118,28
5	5077,17
6	5056,20
7	5096,98
8	5077,17
9	5075,85
10	5096,98
	5057,18

no	Kb/sec
1	3918,56
2	3847,00
3	3822,98
4	3834,95
5	3666,31
6	3893,64
7	3788,24
8	3710,05
9	3968,56
10	4058,16
	3850,85

no	Kb/sec
1	3743,37
2	3666,31
3	3687,71
4	3666,31
5	3765,67
6	3634,35
7	3532,14
8	3788,97
9	3788,24
10	3721,33
	3699,44

mtu 300	
no	Kb/sec
1	3282,42
2	3464,07
3	3231,53
4	3207,20
5	3407,44
6	3150,98
7	3425,91
8	3389,17
9	3198,82
10	3223,56
	3298,11

mtu 300	
no	Kb/sec
1	3512,51
2	3522,61
3	3512,51
4	3522,61
5	3522,61
6	3521,98
7	3532,78
8	3521,98
9	3542,36
10	3532,14
	3524,41

mtu 150	
no	Kb/sec
1	1187,44
2	1129,66
3	1169,72
4	1148,26
5	992,84
6	1197,68
7	1197,68
8	1152,52
9	1158,92
10	1287,39
	1162,21

mtu 150	
no	Kb/sec
1	1204,52
2	1255,22
3	1217,47
4	1338,21
5	1379,48
6	1294,11
7	1296,68
8	1212,72
9	1151,50
10	1279,49

A -3. Nilai Throughput yang didapat pada pengujian MTU pada protokol TCP menggunakan FTP untuk tipe file bervariasi.

Nilai Throughput PC Router Sebelum Dioptimasi mtu 1500	
no	Kb/sec
1	4735,56
2	5009,34
3	5047,98
4	4896,57
5	5157,10
6	5024,94
7	4976,47
8	4959,39
9	4813,59
10	4879,33
	4950,03

Nilai Throughput PC Router Setelah Dioptimasi mtu 1500	
no	Kb/sec
1	5458,2
2	5415,962
3	5472,324
4	5394,884
5	5005,794
6	5013,678
7	5046,082
8	4997,998
9	5231,59
10	5355,85
	5239,24

mtu 1000	
no	Kb/sec
1	4588,12
2	4805,82
3	4976,05
4	4988,65
5	5041,85
6	4894,17
7	4772,99
8	4756,08
9	4878,99
10	4857,98
	4856,07

mtu 1000	
no	Kb/sec
1	4917,414
2	4959,386
3	4978,872
4	4961,732
5	4962,696
6	4926,048
7	4876,116
8	4863,986
9	4835,97
10	4960,852
	4924,31

mtu 500	
no	Kb/sec
1	3576,46
2	3647,38
3	3439,14
4	3590,18
5	3778,23
6	3661,60
7	3584,51
8	3575,30
9	3577,41
10	3748,88
	3617,91

mtu 500	
no	Kb/sec
1	4171,882
2	4093,746
3	4250,404
4	4252,178
5	4186,43
6	4250,404
7	4396,698
8	4089,73
9	4123,932
10	4065,71
	4188,11

mtu 300	
no	Kb/sec
1	3290,58
2	3209,31
3	3301,25
4	3237,36
5	3385,72
6	3320,30
7	3343,56
8	3345,97
9	3407,56
10	3054,10
	3289,57

mtu 300	
no	Kb/sec
1	3479,01
2	3429,436
3	3468,642
4	3474,314
5	3478,904
6	3474,002
7	3542,7
8	3494,468
9	3507,226
10	3517,152
	3486,59

mtu 150	
no	Kb/sec
1	1431,57
2	1413,29
3	1269,24
4	1453,89
5	1422,18
6	1340,84
7	1464,23
8	1421,94
9	1432,54
10	1606,05
	1425,58

mtu 150	
no	Kb/sec
1	1544,812
2	1683,13
3	1657,312
4	1618,068
5	1655,882
6	1615,26
7	1682,842
8	1667,27
9	1660,31
10	1686,494
	1647,14

A -4. Nilai Throughput yang didapat pada pengujian MTU pada protokol UDP menggunakan Iperf.

Nilai Throughput PC Router Sebelum Dioptimasi mtu 1500	
no	Mbit/sec
1	46,2
2	48,2
3	47,6
4	47,6
5	43,3
6	40,3
7	46,0
8	40,1
9	44,0
10	42,2
	44,6

Nilai Throughput PC Router Setelah Dioptimasi mtu 1500	
no	Mbit/sec
1	51,4
2	54,2
3	54,3
4	54,8
5	54,9
6	54,9
7	54,5
8	54,7
9	54,8
10	54,7
	54,3

mtu 1000	
no	Mbit/sec
1	45,1
2	44,0
3	43,7
4	45,1
5	41,7
6	44,1
7	44,2
8	42,4
9	42,6
10	45,9
	43,9

mtu 1000	
no	Mbit/sec
1	48,5
2	48,3
3	48,5
4	48,1
5	48,6
6	47,6
7	48,6
8	48,2
9	48,5
10	47,3
	48,2

mtu 500	
no	Mbit/sec
1	33,1
2	34,2
3	32,7
4	33,6
5	31,2
6	33,1
7	33,8
8	33,3
9	33,7
10	31,6
	33,0

mtu 500	
no	Mbit/sec
1	33,5
2	34,5
3	34,6
4	33,6
5	34,7
6	34,6
7	32,7
8	26,4
9	34,4
10	34,5
	33,4

mtu 300	
no	Mbit/sec
1	25,2
2	24,6
3	24,8
4	25,5
5	24,9
6	24,8
7	25,9
8	25,5
9	25,6
10	25,5
	25,2

mtu 300	
no	Mbit/sec
1	25,9
2	26,4
3	25,3
4	26,3
5	25,4
6	25,2
7	25,9
8	25,3
9	26,3
10	25,6
	25,8

mtu 150	
no	Mbit/sec
1	13,0
2	13,2
3	13,1
4	13,1
5	12,9
6	13,1
7	12,4
8	11,8
9	12,0
10	12,1
	12,7

mtu 150	
no	Mbit/sec
1	13,1
2	13,3
3	13,4
4	10,3
5	13,3
6	13,4
7	13,3
8	13,3
9	13,3
10	13,3
	13,0

A -5. Nilai Throughput yang didapat pada pengujian MTU pada protokol UDP menggunakan Qcheck.

mtu 1500	
no	Kbps
1	1000,436
2	1000,603
3	1000,436
4	1000,436
5	1000,436
6	1000,603
7	1000,603
8	1000,436
9	1000,603
10	1000,436
	1000,503

mtu 1500	
no	Kbps
1	1000,436
2	1000,603
3	1000,603
4	1000,603
5	1000,603
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,603
	1000,520

mtu 1000	
no	Kbps
1	1000,436
2	995,792
3	1000,436
4	1000,603
5	1000,603
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,603
	1000,022

mtu 1000	
no	Kbps
1	1000,603
2	1000,436
3	1000,436
4	1000,436
5	1000,436
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,436
	1000,453

mtu 500	
no	Kbps
1	1000,436
2	1000,436
3	1000,436
4	1000,436
5	1000,603
6	1000,436
7	1000,770
8	1000,436
9	1000,436
10	1000,436
	1000,486

mtu 500	
no	Kbps
1	1000,269
2	1000,603
3	1000,436
4	1000,436
5	1000,436
6	1000,770
7	1000,603
8	1000,436
9	1000,436
10	1000,436
	1000,486

mtu 300	
no	Kbps
1	1000,436
2	1000,436
3	1000,603
4	1000,436
5	1000,770
6	1000,436
7	1000,603
8	1000,436
9	1000,436
10	1000,436
	1000,503

mtu 300	
no	Kbps
1	1000,436
2	1000,436
3	1000,770
4	1000,269
5	1000,603
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,436
	1000,469

mtu 150	
no	Kbps
1	1000,603
2	1000,603
3	1000,436
4	1000,436
5	1000,436
6	1000,436
7	1000,436
8	1000,603
9	1000,436
10	1000,770
	1000,520

mtu 150	
no	Kbps
1	1000,603
2	1000,436
3	1000,603
4	1000,436
5	1000,436
6	1000,436
7	1000,436
8	1000,436
9	1000,436
10	1000,436
	1000,469